UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Bernardo Victor Engelke

SISTEMA DE RECOMENDAÇÃO PARA A PLATAFORMA DSPACE

Florianópolis

2016

Bernardo Victor Engelke

SISTEMA DE RECOMENDAÇÃO PARA A PLATAFORMA DSPACE

Trabalho de Conclusão de Curso submetido ao Curso de Ciência da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Prof. Dr. Roberto Willrich

Florianópolis

2016

Bernardo Victor Engelke

SISTEMA DE RECOMENDAÇÃO PARA A PLATAFORMA DSPACE

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de Bacharel em Ciências da Computação, e aprovado em sua forma final pelo Curso de Ciência da Computação.

Florianópolis, 08 de julho 2016.

Prof. Dr. Renato Cislaghi
Coordenador do Projetos

Banca Examinadora:

Prof. Dr. Frank Siqueira

Prof. Dr. Roberto Willrich
Orientador

Gustavo Tonini MSc.

RESUMO

Repositórios Digitais estão ficando cada vez maiores e significativos dentro da Web, permitindo uma melhor distribuição do conhecimento produzido dentro de instituições ou empresas. Uma das soluções abertas mais adotadas para a implantação de Repositórios Digitais é o DSpace. Esta solução de repositório digital oferece diversas funcionalidades suportando o gerenciamento do fluxo de submissão de conteúdos, armazenamento e distribuição de diversos tipos de recursos digitais. Os tipos de recursos digitais dependem dos objetivos da organização ou pessoa, podendo ser trabalhos acadêmicos, como teses, dissertações, monografias e artigos, ou de caráter mais geral, como vídeos, atas e trabalhos artísticos. Com o crescente número de conteúdos digitais disponibilizados, fica cada vez mais complexa a tarefa dos usuários deste tipo de plataforma localizarem conteúdos de seu interesse. Uma forma usual de apoiar os usuários a localizar conteúdos de interesse é implantar Sistemas de Recomendação. Sistemas de Recomendação, integrados a Repositórios Digitais, aplicam técnicas de filtragem para indicar ao usuário uma lista de conteúdos do repositório com base no perfil deste usuário. Neste sentido, no presente projeto, propôs-se a implantação na plataforma DSpace de um sistema de recomendações utilizando um modelo Web service baseado em tecnologias da Web semântica. A escolha desta solução foi devido à facilidade de instalação destes serviço em repositórios digitais, com pouco impacto no código. A avaliação do impacto de instalação deste sistema de recomendação baseado em Web service no DSpace também é um dos objetivos deste projeto.

Palavras-chave: Repositórios Digitais, DSpace, Sistemas de Recomendação, Web service

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

RD	Repositório Digital
DC	Dublin Core
UFSC	Universidade Federal de Santa Catarina
RI	Repositório Institucional
REST	Representational State Transfer
IBICT	Instituto Brasileiro de Informação em Ciência e Tecnologia
MIT	Massachusetts Institute of Technology
HP	Hewlett Packard
JSPUI	Java Server Pages User Interface
XMLUI	eXtented Mark Language User Interface
RDF	Resource Description Framework
API	Application Programming Interface
CNRI	Corporation for National Research Initiatives
KB	Knowledge Base
BD	Banco de dados
OWL	Web Ontology Language
JSON	JavaScript Object Notation
SAX	Simple API for XML
XML	eXtensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	eXtensible Stylesheet Language for Transformation
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

XHTML	Xtensible Hypertext Markup Language
SoC	Separation of Concerns
PDF	Portable Document Format
SVG	Scalable Vector Graphics
AOP	Aspect-Oriented Programming
SWORD	Simple Web-service Offering Repository Deposit
SO	Sistema Operacional
RAM	Random Access Memory
GB	Gigabyte
TB	Terabyte
JDK	Java Development Kit
JRE	Java Runtime Environment
IDE	Integrated Development Environment
WAR	Web application ARchive
HTTP	Hypertext Transfer Protocol
CPU	Central Processing Unit

SUMÁRIO

1 INTRODUÇÃO

No contexto da Ciência da Informação, existe há muito tempo uma preocupação na preservação de perpetuação de documentos produzidos dentro e fora da esfera digital. Muito se tem pensado em modelos e soluções de armazenamento para que não haja perdas massivas de informações e que elas possam ser expostas livremente ao público. Foi com essa preocupação que surgiu a ideia da criação de plataformas que pudessem armazenar conteúdos digitais a fim de dar uma confiabilidade para a questão de segurança e preservação do armazenamento de mídias em formato digital(ARELLANO, 2004).

O conceito de Repositórios Digitais (RDs) surgiu justamente para sanar essa necessidade de preservação e armazenamento a longo prazo. Conteúdos digitais podem ser disseminados de várias formas. Uma delas é através da criação de RDs, que armazenam coleções de conteúdos digitais, geralmente voltados a um escopo específico. Este tipo de sistema tem tido amplo crescimento ao longo dos últimos anos (CASAGRANDE, 2014).

Apesar de existirem soluções proprietárias, instituições geralmente tendem a usar soluções de código aberto para a implementação de seus RDs. Um expoente na classe desses softwares para a criação de RDs é o DSpace(DSPACE, 2016a).

A estrutura do DSpace é dividida em três principais categorias: Comunidades; Coleções; e Itens. Uma comunidade possui um conjunto de coleções que detém um conjunto de itens. Por sua vez, item contém um grupo de metadados representados pelo padrão Dublin Core (DC)(DCMI et al., 2012) sobre o conteúdo em questão, servindo para sua busca e referência dentro da plataforma(TANSLEY et al., 2003).

O DSpace é um software que visa operar como um serviço institucional centralizador, ou seja, diferentes comunidades inerentes à instituição, como laboratórios, departamentos e unidades administrativas podem ter suas próprias áreas dentro do sistema. Os usuários membros das comunidades podem então, a partir de uma interface Web, depositar seus materiais de interesse(TANSLEY et al., 2003).

Um exemplo de RD usando a plataforma DSpace é o Repositório Institucional da UFSC (`http://repositorio.ufsc.br`). O objetivo deste repositório é aumentar a visibilidade de sua produção acadêmica

para dentro e fora de seu ambiente. Atualmente, o repositório da UFSC conta com mais de 80.000 itens dentro de seu banco de dados. Segundo a própria instituição, o Repositório Institucional (RI) possui como objetivos:

> Contribuir para o aumento da visibilidade dos pesquisadores e da produção científica da UFSC; preservar a memória intelectual da Universidade; reunir em um único local virtual e de forma permanente a produção científica e institucional, disponibilizando o livre acesso aos conteúdos digitais e ampliar e facilitar o acesso à produção científica de uma forma geral.[1]

Segundo Casagrande (2014), é crescente o número de RDs na forma de RIs. Consequentemente, cresce também o tamanho e números de coleções contidas nesses repositórios. Desse modo, devido a esse progressivo número de itens disponíveis, gera-se um problema de sobrecarga de informações fazendo com que o usuário realize um grande esforço para buscar um item de seu interesse.Uma forma de resolver essa questão é na implementação de um sistema de recomendações que registra o acesso do usuário a um item e cria-se uma lista de sugestões de itens com base nos metadados desejados pela implementação (SALLES; WILLRICH, 2015) pois a ferramentas utilizada pela instituição (DSpace) ainda não apresenta tal sistema.

Portanto, no presente trabalho, pretende-se utilizar de uma implementação via Web service como estratégia de solução, que, através desta, fornece-se um maior desacoplamento do sistema de recomendação com o RD, possibilitando inclusive a portabilidade para outras plataformas de conteúdo de preservação digital afins.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste projeto de conclusão de curso é implantar na plataforma DSpace um Sistema de Recomendação. Para tal, é adotado um Web service REST de recomendação proposto por Salles e Will-

[1] Disponível em `https://repositorio.ufsc.br/` acessado em 8 de julho de 2016

rich (2015), que se apoia em tecnologias da Web Semântica e Banco de Dados baseados em grafos para gerar recomendação com base nos perfis dos usuários. Mais especificamente, o objetivo deste projeto é implementar um cliente deste Web service de recomendação integrado na plataforma DSpace. A meta é avaliar a complexidade de integração desta solução de sistema de recomendação na plataforma DSpace. Também serão realizados experimentos visando avaliar o sistema de recomendação proposto no contexto do Repositório Institucional da UFSC.

Figura 1 – Estrutura hierárquica de comunidades, sub-comunidades e coleções do repositório institucional da Universidade Federal de Santa Catarina. Fonte: `https://repositorio.ufsc.br/`

1.1.2 Objetivos Específicos

Além do objetivo principal, procura-se alcançar os seguintes objetivos específicos:

- Obter conhecimentos na área de sistemas de recomendação, e especificamente a solução de Web service de Recomendação proposto por Salles e Willrich (2015);
- Implementar um cliente de Web service na plataforma DSpace que se conecte com um servidor onde situa-se a lógica de negócios do sistema de recomendação;
- Avaliar o impacto na adaptação do repositório DSpace para atuar como um cliente do serviço de recomendação.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizados da seguinte forma: Capítulo 1 procura introduzir os conceitos básicos que serão abordados neste trabalho, dando uma prévia de toda fundamentação teórica do trabalho. No capítulo 2 é apresentada a definição e soluções referentes à repositórios digitais. Em seguida no capítulo 3, aborda-se os sistemas de recomendação, apresentando a proposta para solução da problemática em questão. Por fim, o capítulo 4 trata da implementação e testes da solução proposta, finalizando com a conclusão apresentada no capítulo 5.

2 REPOSITÓRIOS DIGITAIS

Este capítulo apresenta os principais conceitos relacionados a Repositórios digitais, incluindo definições, soluções abertas e aspectos técnicos.

2.1 DEFINIÇÃO

Na análise de Heery e Anderson (2005), os autores relatam uma preocupação em definir um repositório digital. Segundo eles, uma gama crescente de áreas de atividades dentro do contexto de ciência da informação referem-se a coleções de conteúdos depositadas como "repositórios". A fim de incentivar a comunicação entre essas áreas, e promover a interoperabilidade, é preciso definir as características de "repositórios"e buscar a coerência de uma abordagem comum. Para tal, Heery e Anderson (2005), propõem algumas dessas características para diferenciar repositórios digitais de qualquer outra coleção digital:

- O conteúdo precisa ser depositado em um repositório, quer seja pelo criador de conteúdo, proprietário ou terceiro;
- A arquitetura do repositório deve oferecer funcionalidades para gerenciar o conteúdo, bem como os metadados;
- O repositório deve oferecer um conjunto mínimo de serviços básicos, como, por exemplo, inserção, retorno, busca e controle de acesso;
- O repositório deve ser confiável, mantido e gerenciável.

Dentro do cenário brasileiro, O IBICT (Instituto Brasileiro de Informação em Ciência e Tecnologia) (IBICT, 2016) define RDs como:

> Os repositórios digitais (RDs) são bases de dados online que reúnem de maneira organizada a produção científica de uma instituição ou área temática. Os RDs armazenam arquivos de diversos formatos. Ainda, resultam em uma série de benefícios tanto para os pesquisadores quanto

> às instituições ou sociedades científicas, proporcionam maior visibilidade aos resultados de pesquisas e possibilitam a preservação da memória científica de sua instituição. Os RDs podem ser institucionais ou temáticos. Os repositórios institucionais lidam com a produção científica de uma determinada instituição. Os repositórios temáticos com a produção científica de uma determinada área, sem limites institucionais.

2.2 SOLUÇÕES ABERTAS

Para a implementação de RDs como RIs, instituições são levadas a aderir a soluções de código aberto (SALLES; WILLRICH, 2015).

São exemplos de softwares de RDs: DSpace, EPrints, Digital Commons, OPUS e Fedora(OLIVEIRA; CARVALHO, 2011; CASTAGNé, 2013).

O OpenDOAR (Diretório de Repositórios de Acesso Aberto em inglês) registra um uso majoritário (quase 50%) pelo software DSpace como observado na figura 2.

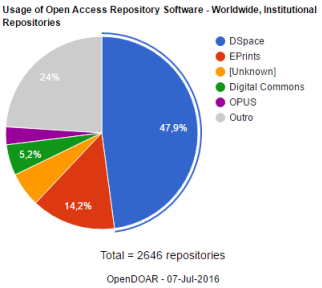

Figura 2 – Estatística de uso de plataformas de RDs com código aberto. Fonte: `http://www.opendoar.org/find.php?format=charts`, acessado em 7 de Julho de 2016.

Levando em contas esses dados, foi escolhido o software DSpace para a realização da implementação da proposta. Portanto, nas próxi-

mas seções, o texto se concentrará em descrever com mais detalhes seu funcionamento e tecnologias usadas.

2.3 DSPACE

Criado em 2002 pelo MIT e os pelos laboratórios da Hewlett Packard (HP), é um software para criação de RDs, em geral RIs, registrando um total de 1356 implementações da plataforma[2], tornando-o assim o software mais usado mundialmente para tal finalidade. É usado por bibliotecas, arquivos e centros de pesquisa(OLIVEIRA; CARVALHO, 2011; CASTAGNé, 2013).

A plataforma DSpace possui uma estrutura baseada em comunidades e coleções, conseguindo assim, criar uma representação dos centros, departamento ou unidades administrativas de uma instituição, podendo abrigar os mais variados formatos de mídias digitais (figura 3)(OLIVEIRA; CARVALHO, 2011).

Dentro de coleções são armazenados os itens que, juntamente com o conteúdo digital, possuem os metadados para sua classificação e indexação(SMITH et al., 2003). A figura 2 procura dar uma visão mais abrangente de todos os objetos que fazem parte da organização dentro do DSpace.

Segundo os autores do projeto, DSpace é um software gratuito de âmbito acadêmico, sem fins lucrativos e para construção de repositórios digitais abertos para organizações. A plataforma preserva e permite acesso fácil e livre para todos os tipos de conteúdos digitais, incluindo textos, imagens, imagens em movimento, mpegs e conjuntos de dados. Possui uma crescente comunidade de desenvolvedores que ajudam a expandir e melhorar ainda mais o software para cada versão lançada(DSPACE, 2016a).

O projeto ainda afirma que o DSpace é um software de fácil instalação, sendo ele completamente customizável para caber nas necessidades de qualquer organização.

[2]Dados retirados do ROAR em julho de 2013 (Registro de Repositórios de Acesso Aberto em inglês) - `http://roar.eprints.org/`

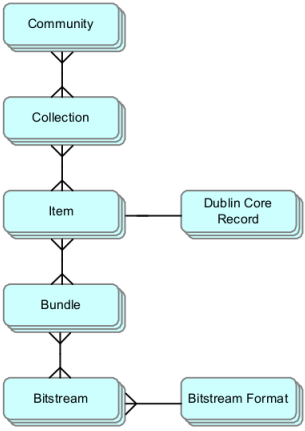

Figura 3 – Visão geral da estrutura hierárquica dos objetos no DSpace. Fonte (TANSLEY et al., 2003)

2.4 DETALHES TÉCNICOS DO DSPACE

2.4.1 Metadados

DSpace faz o uso do padrão Dublin Core para descrição de metadados de seus itens, havendo um mínimo de três campos requeridos para a composição de um item: título; língua; e a data de submissão. Todos os outros campos, como resumo, palavras-chaves, detalhes técnicos, direitos autorais, entre outros, são opcionais. Estes metadados são visualizados dentro de um item no DSpace(SMITH et al., 2003).

2.4.2 Interface

A interface disponível para usuários é *web-based*, contendo várias sub-interfaces. Uma para submetedores de itens e/ou qualquer processo que esteja envolvido com submissão; uma para usuários finais que buscam pela informação; e outra para administradores do sistema(SMITH et al., 2003).

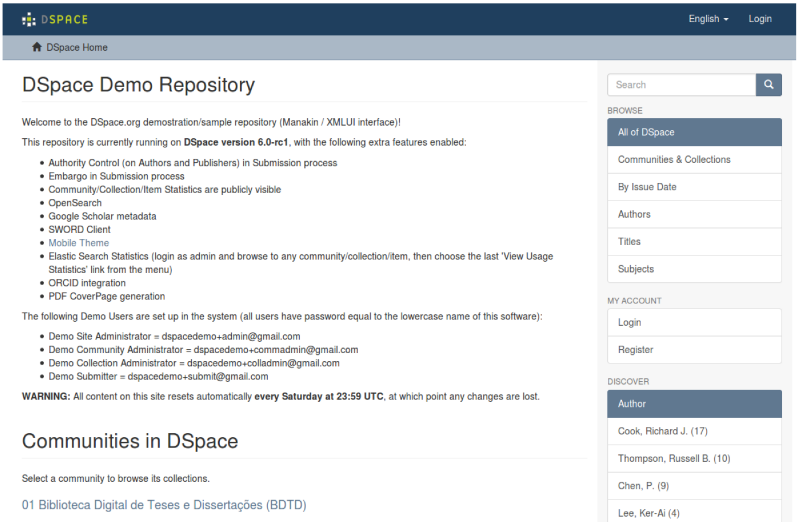

Figura 4 – Página inicial em interface XMLUI do DSpace.

A interface para usuários finais, ou interface pública, contém campos de busca com retorno de valores para navegação ou valores de metadados. Uma vez que o item é localizado no sistema, o usuário é capaz de fazer o download do arquivo em anexo (se disponível e se houver) através de seu navegador web(SMITH et al., 2003).

Ainda a respeito da interface web, ela pode ser dividida em duas opções de implementação, a JSPUI (Java Server Pages User Interface) e XMLUI (eXtented Mark Language User Interface), com vantagens e desvantagens de cada uma delas, cabendo ao administrador ou equipe responsável pelo repositório decidir qual delas utilizar(SHINTAKU; MEIRELLES, 2010).

A versão atualmente utilizada do XMLUI no DSpace é chamada de Manakin. Essa versão passa a fazer uso do framework Apache Cocoon(`http://cocoon.apache.org/`) para usar uma abordagem Simple API for XML (SAX). Para completar essa customização, Manakin usa temas(*Themes*) para estilo de conteúdo e pacotes chamados de *Aspects* para modularizar a geração do conteúdo(PHILLIPS et al., 2005).

Não estando presente inicialmente, a interface XMLUI apareceu a partir da versão 1.5 provendo grandes melhorias(SHINTAKU; MEIREL-

LES, 2010). A principal delas é a possibilidade de customização, oferecendo ao usuário a escolha de adotar características distintas para comunidades, coleções e itens. Outras inovações são a separação da camada de negócio da interface, internacionalização e localização de conteúdos e compatibilidade com a então atual interface baseada em JSP(WOLF; MONTEIRO; VALMORBIDA, 2013).

Como desvantagens, podemos citar a maior complexidade para alteração e manutenção. Como utiliza folha de estilos e programas conversores (XSL), exige profissionais mais especializados e de maior conhecimento em edição de *front-end* de sistemas web(SHINTAKU; MEIRELLES, 2010).

A figura 4 mostra a interface Manakin XMLUI, à qual foi aplicado o tema Mirage, padrão da instalação da versão 6.0.

A arquitetura do Manakin consiste de 3 camadas. Em cada camada, níveis de componentização estão presentes, possibilitando um desenvolvimento em paralelo. Progressivamente, é requerido menor experiência para desenvolvimento, conforme desce-se nas camadas(PHILLIPS et al., 2005).

- Camada de desenvolvimento Java/Cocoon: Requer conhecimento em Java e Cocoon framework e possibilita realizar qualquer customização funcional para o DSpace;
- Camada de tema XML/XSL: Exige conhecimento XML e XSLT, embora não precisando mais da linguagem Java. Nesta camada, informações processadas pelo DSpace podem ser manipuladas antes de mostrar ao usuário final;
- Camada de estilos HTML/CSS: Necessita apenas de conhecimento XHTML básico e permite modificar o estilo das informações que são mostradas diretamente para o usuário final.

2.4.2.1 Apache Cocoon

Cocoon descreve-se como um framework para desenvolvimento web construído sob os conceitos de *Separation of Concerns* (SoC) e arquiteturas baseadas em componentes. Juntando com a API SAX, estas características reúnem argumentos favoráveis necessários para uso desta tecnologia(PHILLIPS et al., 2005).

O princípio do SoC habilita o desenvolvimento em paralelo das aplicações. Cada desenvolvedor pode concentrar-se em uma única parte do projeto sem alteração ou influência de outro. Como o desenvolvimento do DSpace é largamente provido por contribuições de comunidades de programadores, o uso do SoC se veio como uma facilidade, pois antes haviam muitos conflitos de projetos decorrentes das amarrações que interfaces JSP possuiam(PHILLIPS et al., 2005).

2.4.2.2 *Sitemap*

O *sitemap* é um cojunto de documentos XML que descreve a configuração de todos os componentes Cocoon. O *sitemap* contém duas principais partes: a definição da seção dos componentes que descreve cada tipo de componente, e uma seção de pipeline que define como estes componentes são arranjados. Pode-se dizer que o *sitemap* é o coração do framework(PHILLIPS et al., 2005).

2.4.2.3 Temas

Como já dito, os temas contribuem para a estilização do conteúdo gerado e disponível pelo Manakin. O formato é tipicamente em XHTML, entretanto, não existe impedimento para uso de outros formatos como PDF ou SVG, por exemplo. Temas são implementados como folhas de estilo XSL. Todos os conteúdos digitais usados para a estilização do tema são empacotados juntos para criar assim uma portabilidade. Temas podem ser configurados para ser aplicados à uma comunidade/coleção particular ou em muitas ao mesmo tempo. Também podem ser aplicados em uma única página arbitrária(PHILLIPS et al., 2005).

2.4.2.4 *Aspect*

No paradigma de programação orientada a aspectos (Aspect-Oriented Programming (AOP)), programas são separados em partes distintas sobrepondo-se o mínimo possível. Estas partes são chamadas de *aspects*, e entrelaçam-se para formar o programa. No Manakin, estes

aspects se combinam para formar todas as suas características. Essa é mais uma estrutura que potencializa a interface, pois todos os aspectos estão estruturados separadamente(PHILLIPS et al., 2005).

A ordem com que os aspectos são executados é determinado pelo arquivo de configuração. *aspects.xml*. Nesse caso, aspectos presentes na configuração são empacotados juntos com o código-fonte, *sitemaps* e outros recursos que possam ser requeridos. Este empacotamento faz com que gere-se portabilidade, sendo fácil transportar de um DSpace para outro causando um mínimo de conflito(PHILLIPS et al., 2005).

Em sua versão atual, os aspectos estão disponibilizados da seguinte forma(DURASPACE,):

- ViewArtifacts: Responsável pela exibição de um item individual;
- BrowseArtifacts: Reponsável pela exibição de diferentes opções da navegação;
- SearchArtifacts: Responsável pela exibição de diferentes campos de busca (não usado por padrão);
- Administrative: Responsável pelo conjunto de ferramentas administrativas que o sistema contém;
- E-Person: Responsável pela adminsitração de usuários e todo seu contexto;
- Submission: Responsável pela submissão de novos itens no DSpace;
- Statistics: Responsável pela exibição das estatísticas do DSpace;
- Workflow: Responsável pelas tarefas de fluxo de trabalho que o DSpace apresenta (não usado por padrão);
- XMLWorkfflow: Adicionado a partir da versão 1.8 e substitui desde então o aspecto Workflow;
- Discovery: Também adicionado posteriormente, substitui os aspectos relacionados a buscas até então;
- SwordClient: Referente ao sistema SWORD presente no DSpace e que não será discutido neste trabalho;
- XMLTest: Aspecto criado para dar assitência ao desenvolvedores contendo opções de debugging;

- ArtifactBrowser: Aspecto depreciado. Foi divido em ViewArtifacts, BrowseArtifacts e SearchArtifacts a partir da versão 1.7.

2.4.3 Tecnologias usadas

DSPace foi desenvolvido para ser código aberto, de forma que instituições e organizações possam usufruí-lo com uma quantidade mínima de recursos. O sistema é desenvolvido para rodar em plataforma UNIX abrangendo várias outras ferramentas e middlewares escritos pelos desenvolvedores(SMITH et al., 2003).

Todo o código original é escrito na linguagem de programação Java. Outras partes essenciais para o sistema incluem o sistema de base de dados relacional PostgreSQL, um servidor web (Apache), um servidor Java (Tomcat) e outras bibliotecas de sistema como Jena (RDF) e OAICat(SMITH et al., 2003).

Vale ressaltar que enquanto DSpace é um software aberto e gratuito, seus desenvolvedores (MIT e HP) não oferecem suporte formal para seus usuários. É assumido que instituições que decidirem usar o software possuam recursos suficientes para a manutenção e configuração deste(SMITH et al., 2003).

Os softwares necessários para a execução e instalação do DSpace serão apresentados e discutidos de forma mais aprofundada no capítulo 4.

2.4.4 Requisitos de Hardware

De acordo com o manual do DSpace (DSPACE, 2016b), a recomendação varia muito dependendo do tamanho que o repositório pretende atingir. Para uma aplicação mínima do software, requer-se, 2-3GB de RAM e 20GB de armazenamento, enquanto que para uma aplicação pesada pede-se 8GB de RAM com 1TB de armazenamento.

2.4.5 Sistema Operacional

O DSpace pode ser implantando tanto no Sistema Operacional *UNIX-like* ou Microsoft Windows. Em sistemas operacionais *UNIX-like* já se contém algumas dependências para a instalação pré-instaladas ou que são facilmente instaladas via atualizações em seus repositórios, diferentemente do Microsoft Windows onde todos os programas precisam ser instalados manualmente(DSPACE, 2016b).

2.4.6 Arquitetura de sistema

A arquitetura do DSpace é dividida em 3 camadas: armazenamento, lógica de negócios e aplicação. A camada de armazenamento é implementada usando um sistema de arquivos e gerenciada pelas tabelas do banco de dados do PostgreSQL. A de lógica de negócios é onde as funcionalidades específicas do DSpace residem, incluindo o *workflow*, a gerência de conteúdo, a administração, módulos de busca e navegação. Cada módulo dispõe de uma API que permite que os usuários possam ter as funcionalidades que desejam. Por fim, a camada de aplicação cobre toda a interface do sistema(SMITH et al., 2003).

A figura 5 ilustra a arquitetura do sistema do DSpace.

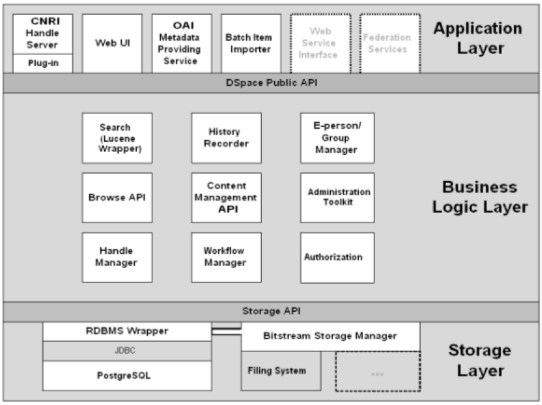

Figura 5 – Arquitetura de sistema do DSpace. Fonte (SMITH et al., 2003)

2.4.7 *Handles*

Um dos objetivos da preservação digital é possibilitar que se busquem e achem conteúdos depositados nos repositórios digitais a longo prazo. Para isso, é crucial que itens citados encontrados em artigos, por exemplo, possam ser encontrados efetivamente por longos períodos de tempo. Para esse fim, DSpace escolheu implementar CNRI Handles (`https://www.handle.net/`) como identificadores de persistência de cada item o qual embui-se das tarefas de gerências de itens de repositórios cadastrados ao redor de mundo(SMITH et al., 2003).

Mais especificamente, no DSpace, Handles são associados a comunidades, coleções e itens. O Conteúdo digital anexado ao item não possui Handle, uma vez que esses podem ser mudados devido a alguma atualização, por exemplo(DSPACE, 2016b).

3 SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação dentro do contexto de RDs têm como função recomendar obras, baseados em um conjunto de informações específicos, que neste caso, advém de metadados de um item pertencente ao RD, e que sejam potencialmente interessantes ao usuário. De modo geral, um sistema de recomendação se utiliza de três operações: o perfil do usuário, que manterá informações necessárias à identificação das preferências do usuário; a filtragem de informação, que seleciona os itens relevantes ao usuário utilizando para isso os perfis de usuário e dados de todos os itens; e o ranqueamento dos itens, servindo para listar de forma ordenada por relevância as obras escolhidas pelo sistema de recomendação(SALLES; WILLRICH, 2015).

3.1 TIPOS DE SISTEMAS DE RECOMENDAÇÃO

3.1.1 Abordagem clássica

Conforme descrito no livro *Recommender Systems Handbook* (RICCI et al., 2011), há 6 tipos de abordagens clássicas de sistemas de recomendações:

- Baseado em Conteúdo: O sistema aprende a recomendar com base em itens similares na preferência do mesmo usuário no passado;
- Filtragem Colaborativa: Como o nome sugere, a recomendação é calculada com base no que outros usuários preferiram no passado;
- Demográfica: Retorna itens baseados no perfil demográfico do usuário. Por exemplo, usuários recebem recomendações de itens baseados em sua língua ou país;
- Baseado em Conhecimento: Recomenda itens baseados em um domínio específico de conhecimento sobre como certas características de itens satisfazem as necessidades e preferências do usuário;
- Filtragem Híbrida: Combina duas ou mais técnicas de sistema de recomendação.

3.1.2 Baseado em grafos

O objetivo da representação em grafos é a otimização do processo de recomendação. No caso da Filtragem Colaborativa, por exemplo, pode-se modelar a recomendação na forma de um grafo bipartido onde os nodos representariam usuários e itens com arestas interligando-os. Em uma técnica baseada em conteúdo, os nodos de um grafo bipartido podem representar itens e valores com ligações entre eles na forma de arestas. Por último, em uma filtragem híbrida, é possível criar um modelo com nodos representando usuários, itens e metadados em um grafo, dessa vez, tripartido. A figura 6 mostra uma ideia dessa último modelo(SALLES; WILLRICH, 2015).

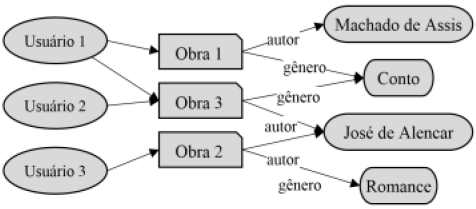

Figura 6 – Recomendação em filtragem híbrida em modelo de grafo. Fonte Salles e Willrich (2015)

3.1.3 Baseado em ontologias

> Uma ontologia é uma representação formalizada e não ambígua do conhecimento de um domínio, definindo conceitos, propriedades e relacionamentos(SALLES; WILLRICH, 2015).

Dessa maneira, seu uso em sistemas de recomendação consiste em explorar o relacionamento entre conceitos e a semântica das propriedades, podendo obter interesses de usuários que não são diretamente observados. Para exemplificar, pode-se citar um usuário que acessa várias obras literárias de autores de um determinado estado. Esse comportamento pode indicar que este usuário pode ter interesses por obras

advindas desse estado. Em sistemas de recomendações tradicionais, esse comportamento não é percebido, pois concentra-se apenas em propriedades de itens já acessados(SALLES; WILLRICH, 2015).

3.2 TRABALHOS RELACIONADOS

Existem diferentes soluções de recomendações para repositórios digitais. Três delas merecem destaque: frameworks de recomendação, add-ons para RDs; e Web service (SALLES; WILLRICH, 2015).

Na primeira, no framework, os mais populares são *Apache Mahout*, *LensKit* e *MyMediaLite*. O uso de framework de recomendação integrados a RDs permite o reuso de componentes de software escaláveis e eficientes para recomendação. No entando, usar frameworks resulta em uma implementação de recomendação bastante acoplada com à estrutura de dados dos RDs, além de ser uma solução específica para um RD, não possibilitando sua generalização e transporta para RDs federados.

A segunda solução , por *add-ons*, já é uma forma de reduzir o esforço de implantação de serviços de recomendação em RDs abertos. No caso específico do DSpace, um trabalho relacionado nesse sentido é o *add-on* proposto em (ELLIOTT; RUTHERFORD; ERICKSON, 2008). Neste trabalho, os autores apresentam o Quambo, que adiciona contexto de pesquisas, itens favoritos e um próprio sistema de recomendação. Porém como se trata de um *add-on*, esta alternativa limita o reuso em RDs de uma determinada solução e assim como a solução em framework, não oferece suporte a RDs federados. Sua implementação é feita toda dentro da estrutura do DSpace deixando claro seu alto acoplamento com o sistema.

Por último, a solução por Web service apresenta a estrutura de maior desacoplamento com os RDs em si, e portanto se mostra a melhor opção em caso de RDs federados pois os dados e usuários podem ser combinados pelo servidor. Para esta solução, podemos citar os trabalhos de Anjorin et al. (2012), Beham et al. (2010) e Lemos et al. (2012). Em Lemos et al. (2012) os autores propõem um sistema de recomendação de fotos baseado em *RestFul Web Service*, permitindo que dispositivos móveis de baixa capacidade de processamento possam usar o serviço. No entando o trabalho visa apenas um único domínio de aplicações (fotos), e os dados utilizados para a recomendação são obtidos de bases de fotos específicas. Em Anjorin et al. (2012), os auto-

res propõem um sistema de recomendação de objetos de aprendizagem possibilitando o seu uso em plataformas de ensino. Mas, novamente, o domínio se limita a apenas objetos de aprendizagem. Em Beham et al. (2010), é introduzido o sistema APOSTDL, que publica recomendações de especialistas. Para isso, o sistema se baseia em uma ontologia de domínio que representa tarefas em um domínio de aprendizagem. O perfil do usuário é implicitamente capturado observando as tarefas realizadas pelo usuário (SALLES; WILLRICH, 2015).

3.3 SISTEMA DE RECOMENDAÇÃO ADOTADO

Em seu trabalho,Salles e Willrich (2015) estudam várias propostas já feitas como embasamento para a sua, e diferentemente dos trabalhos descritos na seção anterior, esta, independente de domínio, possibilitando especificar os conhecimentos do domínio relacionado aos itens a recomendar via a especificação de uma base de conhecimento ontológica.

Portanto, neste serviço planejado, o sistema de recomendação proposto por Salles e Willrich (2015), são identificados quatro principais atores participantes (ilustrados na Figura 7):

- Administrador do contexto: Responsável por cadastrar o contexto (dados usados pelo serviço) no qual o serviço de recomendação será oferecido e de especificar a base de conhecimento (Knowledge Base (KB)) inicial;
- Usuário: São os usuário que acessam o repositório digital e recebem recomendações
- Repositório Digital: Atua como cliente do serviço de recomendação. Tem por função notificar acesso realizado pelos usuários e solicitar a recomendação ao sistema via Web service;
- Serviço Web de recomendação: Serviço de recomendação propriamente dito. Formado pelo Gestor de contexto e Recomendador que implementa os algoritmos de recomendação. Dentro do servidor, as informações são mantidas em Banco de Dados chamados de BD de contextos que contém os dados gerais dos contextos já cadastrados. BDs dos contextos mantém as bases de conhecimento dos contextos já cadastrados.

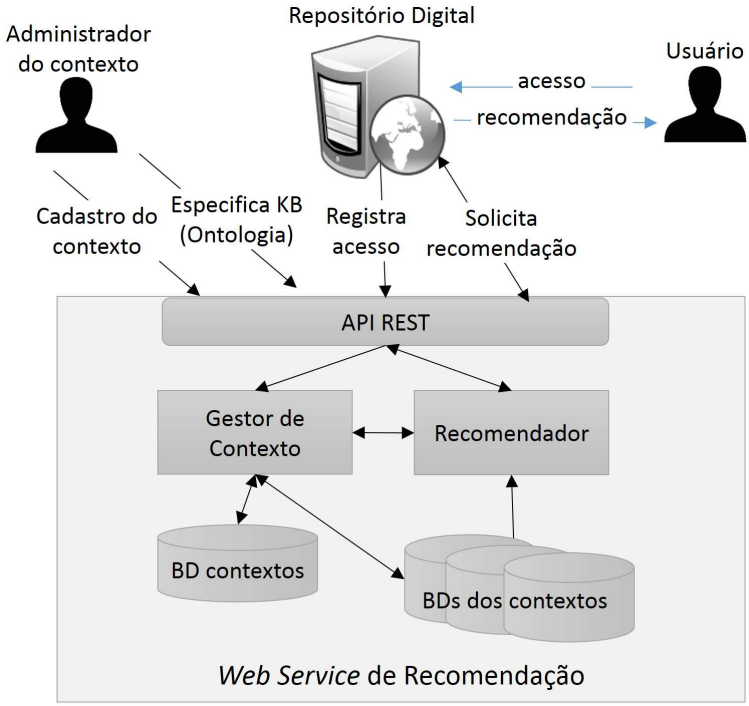

Figura 7 – Interação dos atores no sistema.

3.3.1 Modelagem conceitual dos Dados

Para especificar a base de conhecimento, foi adotada a linguagem OWL (*Web Ontology Language*) (GROUP, 2012). Com isso, descreve-se então uma ontologia de aplicação, uma ontologia de domínio do contexto e um ontologia dos indivíduos. A ontologia de domínio do contexto especifica conceitos e relacionamentos associados ao domínio de conhecimento relativos aos conteúdos digitais de um contexto de recomendação. Como exemplo de ontologia de domínio de contexto, podemos citar o domínio de obras literárias, onde obras, autores e gêneros literários representam essa ontologia(SALLES; WILLRICH, 2015).

A ontologia de aplicação é um ontologia desenvolvida para uma aplicação específica. Portanto, para este trabalho, é criada uma ontolo-

gia de aplicação chamada de **RecOnt**, ou ontologia de recomendação. A figura 8 apresenta a ontologia de recomendação. A *Audience* representa o grupo de usuários, que é o foco da recomendação. *Item* significa o item a ser recomendado. E o *Factor of Interest*, simboliza determinado conceito importante a Item. No caso do DSpace com o padrão *Dublin Core*, um conceito relevante poderia ser um grupo de autores e palavras-chaves de um mesmo item(SALLES; WILLRICH, 2015).

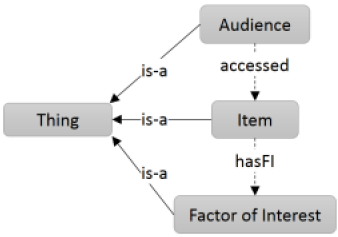

Figura 8 – Ontologia de recomendação RecOnt(SALLES; WILLRICH, 2015).

É importante observar que esses conceitos descritos são representados por URIs, fazendo com que o sistema de recomendação seja desacoplado do repositório digital, no caso deste trabalho, do DSpace. Assim, o sistema provê privacidade ao usuário, pois não captura nenhuma informação pessoal sobre o tipo de conteúdo que ele está acessando. Finalmente, podemos dizer que a presente proposta se torna uma solução de propósito geral, já que se adapta facilmente a qualquer domínio do conhecimento(SALLES; WILLRICH, 2015).

3.3.2 Interface de Acesso

A interface de acesso do serviço é via Web service do tipo REST, disponibilizando dados em formato JSON. Escolheu-se este tipo por se tratar de um formato simples e de fácil interpretação e manipulação por humanos e máquinas independente de linguagem de programação e/ou framework e sistema operacional, já que os dados precisarão passar por algum tipo de processamento, não só Web como localmente no servidor(SALLES; WILLRICH, 2015).

3.3.3 Banco de dados orientado a grafos

Os dados coletados pelo sistema de recomendação são armazenados e organizados em forma de grafo em um banco de dados orientado a grafos. Como sistemas de recomendação estão relacionados à escalabilidade e considerando resultados promissores de testes já realizados, optou-se pelo uso do banco de dados Neo4j para tal finalidade(SALLES; WILLRICH, 2015).

3.4 CONSIDERAÇÕES FINAIS

Embora o foco do projeto seja o software DSpace, havendo então outras maneiras de construção de sistemas de recomendações que não a proposta neste trabalho, por exemplo, a utilização de *add-ons* baseados em trabalhos anteriores (ELLIOTT; RUTHERFORD; ERICKSON, 2008), a estratégia escolhida é a de um sistema de recomendações em Web service baseado em ontologias. Desta maneira, é possível oferecer um baixo acoplamento do sistema como um todo, pois toda a lógica de negócio da recomendação será computada em um servidor diferente, cabendo ao RD apenas atuar como cliente, simplesmente conectando-se ao serviço já mencionado. Além disso, a solução proposta oferece formas de prover a recomendação em federações, pois os dados e usuários podem ser combinados pelo servidor(SALLES; WILLRICH, 2015).

4 SISTEMA DE RECOMENDAÇÃO PARA DSPACE

Este capítulo descreve a implantação do sistema de recomendação baseado em Web service no DSpace . A seção 4.1 descreve os componentes existentes do sistema bem como suas especificações usadas para a implantação, enquanto que a seção 4.2 atenta em descrever como são definidas as ontologias que comporão o sistema de recomendação. A seção 4.3 visa especificar a forma que o sistema atuará no DSpace, subseguindo com a seção 4.4 e 4.5 que se ocupará da parte de implementação e avaliação da integração respectivamente.

4.1 COMPONENTES DO SISTEMA DE RECOMENDAÇÃO

Os principais componentes do Sistema de Recomendação proposto são (Figura 9): cliente, servidor do repositório e serviço de recomendação.

Figura 9 – Principais componentes do sistema de recomendação proposto

4.1.1 Cliente

No sistema, o cliente é responsável pelo registro e acesso de recursos disponíveis no repositório. Para tal, sua atuação utiliza-se somente

de um navegador para acesso à plataforma DSpace.

4.1.2 Repositório DSpace

Neste caso, o Repositório é constituído da plataforma para repositórios digitais DSpace. A versão do software utilizada neste projeto foi a 6.0 e sua instalação obedeceu os passos e instruções descritos nesta seção.

A máquina usada para o desenvolvimento foi um Intel Core i5 com 16GB de RAM e disponibilidade de 500GB de armazenamento. O sistema operacional (SO) adotado nesta máquina foi a distribuição Linux Ubuntu na versão 14.04.

Como etapa preliminar a instalação do DSpace propriamente dito, foi necessária a instalação dos seguintes pacotes de softwares (DSPACE, 2016b):

- *Java SE Development Kit*: A documentação do DSpace indica as opções de máquina virtual Oracle Java JDK 1.7+ ou OpenJDK 1.7+. Neste projeto foi adotada a implementação livre a gratuita do Java OpenJDK $1.8.0_1 11$;
- Apache Maven: Maven é necessário no primeiro estágio de compilação do DSpace. A utilização do Maven permite criar módulos no código-fonte, fazendo com que se possa modificar e compilar partes do código-fonte sem a necessidade de construir todo ele novamente para cada alteração feita. Neste projeto foi utilizado a versão 3.0.5 do Maven. Até a presente versão do DSpace (6.0), os módulos criados pelo Maven são: dspace-api; dspace-jspui; dspace-oai; dspace-rdf; dspace-rest; dspace-services; dspace-solr; dspace-sword; dspace-swordv2; dspace-xmlui; e dspace-xmlui-mirage2;
- Apache Ant: Apache Ant é requerido como segundo estágio do processo de compilação. Ant é usado uma vez que os pacotes são construídos pelo Maven no diretório *[dspace-source]/dspace/target/dspace-installer*. Neste projeto foi instalado o Apache Ant 1.9.3;
- Sistema Gerenciador de Banco de Dados: A Documentação do DSpace indica a possibilidade de utilizar o PostgreSQL 9+ ou o Oracle 10+ (nos dois casos sendo necessário a habilitação do

suporte à Unicode). Neste projeto foi adotado o SGBD PostgreSQL. A partir da versão 6.0 do Dspace requer uma versão maior ou igual que a 9.4 com a extensão pgcrypto instalada. Portanto, para atingir estes requisitos, usou-se a versão 9.4.10;

- Apache Tomcat: A versão adotada neste projeto foi a Tomcat 7.0.52, sendo que a documentação do DSpace recomenda a utilização da versão a partir da 7.0.30, visto que versões abaixo desta apresentam problemas de estouro de memória, exigindo grandes quantidade de uso de memória somente para o Tomcat.

Após a configuração dos softwares requeridos no sistema operacional, foram realizados os seguintes passos para instalação do DSpace, conforme o manual DSpace (2016b):

- Criação de um usuário DSpace: **useradd -m [usuário]** (Este usuário precisa ser o mesmo que o Tomcat irá usar);
- *Download* da versão 6.0 do DSpace disponível em seu website;
- Criar usuário e banco de dados em um sistema de gerenciamento de banco de dados escolhido;
- Configuração das variáveis iniciais do sistema disponíveis no arquivo *build.properties*;
- Criação do diretório com permissão de usuário dspace (ou outro nome escolhido) na criação do usuário com base no caminho escolhido no passo anterior;
- Uso do comando **mvn package** dentro do diretório raiz do código-fonte do DSpace para compilação dos módulos. (Este comando utiliza o arquivo *pom.xml* para compilação);
- Após o primeiro passo da compilação, executou-se o comando **ant fresh_install** dentro do diretório *dspace-installer* criado em *[dspace-source]/dspace/target/dspace-installer* para a instalação dos módulos compilados pelo maven no sistema operacional;
- Realização do *deploy* da interface escolhida, disponível no diretório *webapps* do DSpace instalado, no Tomcat;
- Execução do Tomcat e acesso à URL configurada para acessar o sistema.

Conforme discutido anteriormente, existem duas opções para a implementação das interfaces Web do DSpace: JSPUI (Java Server Pages User Interface) e XMLUI (eXtented Mark Language User Interface). Neste projeto foi adotada o XMLUI, pois por razões já enunciadas no capítulo 2, ela apresenta vantagens quando leva-se em questão sua customização.

É relevante destacar que este projeto visa incorporar a funcionalidade de Sistema de Recomendação no DSpace, e não uma simples personalização das interfaces Web (por exemplo, visando atender as identidades visuais de uma determinada organização). A incorporação de novas funcionalidades no DSpace foi um desafio a parte, devido à carência de documentação disponíveis aos desenvolvedores.

Para incorporar as funcionalidades de um Sistema de Recomendação no DSpace, é necessário o uso de sua API e obedecer ao padrão de projeto adotado pelo DSpace. Assim, uma página de visualização necessitará de um novo *aspect* da interface possuindo métodos com assinaturas específicas. Aos que se pode observar, temos os métodos *addBody*, *addOptions*, *addPageMeta*, e *addUserMeta*. Cada método não é excludente ao outro, podendo todos estarem presentes ou não. Esses métodos são responsáveis por receber objetos de um determinado segmento das informações que serão processadas pelas camadas posteriores.

Para a construção da lógica de negócio da solução proposta, ou seja, o tratamento dos dados vindos do DSpace e enviados para o sistema de recomendação, a classe no DSpace pode ser construída sem a adição desses métodos mencionados no parágrafo anterior, estando amarrada somente à sua API.

4.1.3 Serviço Web de Recomendação

É a aplicação que executa a lógica de negócio da recomendação. Seus algoritmos não serão discutidos neste trabalho devido à sua complexidade.

O serviço foi desenvolvido na linguagem Java e utilizado a OWL-API para a importação das ontologias.

Seu hardware é formado por um servidor HP ProLiant DL180G5, processador Intel Quad Core Xeon E5404, 8GB de memória RAM e

500GB de armazenamento. O SO em questão é a distribuição Linux Ubuntu na versão 14.04.

Os softwares instalados e necessários para seu funcionamento são:

- Apache Tomcat 7.0.52;
- Java Oracle 1.8.0_91b14;
- Neo4J 2.3.7.

4.2 ONTOLOGIA DE DOMÍNIO E DE CONTEXTO

Para a instalação da RecOnt no domíno de repositórios DSpace, deve-se definir uma ontologia de domínio e uma ontologia de contexto. Esta seção ilustra como são definidas estas ontologias para implantação do sistema de recomendação em uma instalação DSpace.

A ontologia de domínio define conceitos gerais sobre o domínio dos recursos disponibilizados no repositório DSpace. Para este trabalho, adotou-se uma ontologia de domínio genérica, considerando conceitos genéricos existentes em repositório DSpace, chamada *DCResource*. A Figura 10 apresenta a *DCResource*. Os conceitos desta ontologia são:

- *Resource*: Representa um recurso no DSpace, sendo um recurso digital como um artigo, tese, dissertação, etc. Um *Resource* possui um ou mais autores (*hasAuthor*) e uma ou mais palavras-chave (*hasSubject*);
- *Author*: Representa um autor do recurso;
- *Subject*: Representa uma palavra-chave associada ao recurso.

As classes *Subject* e *Author* foram escolhidas pois os autores deste trabalho consideram estes os metadados DC mais relevante para capturar as preferências/interesses do usuário de um repositório DSpace que armazene qualquer tipo de recursos.

A ontologia de contexto adotada é DCContext, apresentada na Figura 11. Ela define que o item a recomendar é da classe *Resource*, e que os fatores de interesse são *Author* e *Subject*

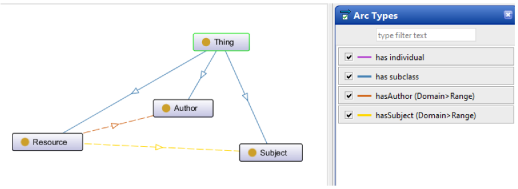

Figura 10 – Ontologia DCResource.

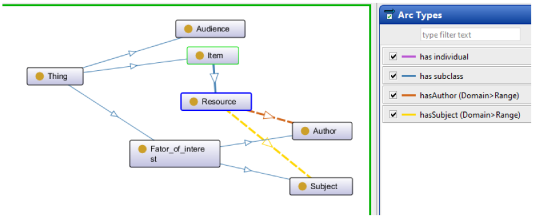

Figura 11 – Ontologia DCContext.

4.3 ESPECIFICAÇÃO DO SISTEMA DE RECOMENDAÇÃO INTEGRADO AO DSPACE

Esta seção visa especificar, utilizando diagramas UML, os locais e formas de recomendação a serem integradas no repositório DSpace.

4.3.1 Registro de Acessos a Recursos

O sistema RecOnt disponibiliza alguns algoritmos de recomendação. A recomendação por popularidade não requer a identificação do usuário acessando o recurso. Já na recomendação por filtragem colaborativa e baseada em conteúdo implementada atualmente exigem a captura do histórico do usuários. Este histórico contém os recursos acessados pelo usuário, bem como os seus metadados.

O Diagrama de Sequência da Figura 12 apresenta o acesso de um recurso por parte de um usuário logado. O método de registro de acesso implementado deve realizar as seguintes ações:

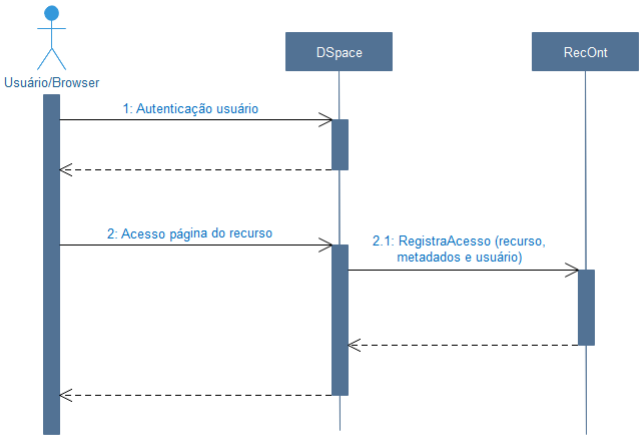

Figura 12 – Diagrama de sequência de um registro de um recurso baseado em conteúdo

- Coletar o identificador do usuário no DSpace;
- Coletar os metadados do recursos acessados considerados relevantes para capturar as preferências do usuário. Neste projeto,considerou-se os metadados autor (dc.contributor.author) e assuntos (dc.subject);
- Representar o item e seus metadados em OWL com base na ontologia de contexto DCContext;
- Chamar o serviço web de registro de acesso da RecOnt.

No caso de um acesso a um recurso por um usuário não logado, o procedimento de registro de acesso é o mesmo, com exceção que é utilizado um usuário especial, chamado *Anonymous*, apenas para fins de recomendação por popularidade.

A requisição do Web service RecOnt para registro de acesso a recursos deve seguir o seguinte padrão:

- **Método POST**:

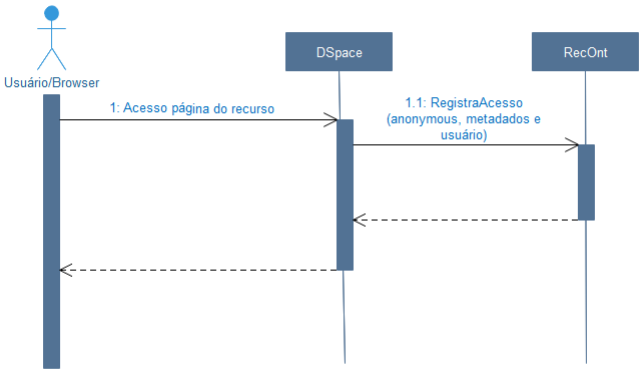

Figura 13 – Diagrama de sequência de um registro de um recurso baseado em popularidade

Exemplo: `http://localhost:8080/{ApplicationName}/{WebService}/access/apiKey/Context/actionTime`

Onde temos os seguintes parâmetros para a construção da estrutura da URL como no exemplo acima:

- ApplicationName: Nome da aplicação;
- WebService: Classe contendo os métodos do Web service;
- access: Método que será executado.
- apiKey: Token responsável pela identificação do repositório cliente;
- Context: Identificação do contexto do repositório, podendo ser 1 ou mais;
- actionTime: Tempo em que o acesso ocorreu.

4.3.2 Apresentação da Recomendação

Existem dois cenários de apresentação da recomendação para os usuários do repositório. O primeiro é para o caso de usuários não autenticados no sistema, e o segundo, para usuários autenticados.

O Diagrama de Sequência da Figura 14 apresenta o acesso à página principal do DSpace por parte de um usuário não logado, ou seja, um usuário anônimo. O método de recomendação implementado deve realizar as seguintes ações:

- Solicitar a recomendação por Popularidade ao RecOnt, utilizando o WebService RecOnt. Conforme previsto na API, esta chamada deverá conter o identificador do usuário *Anonymous* e o identificador do tipo de recomendação solicitada;
- Receber a lista de recursos recomendados na forma de uma estrutura JSON. Cada recurso será identificado pelo seu handle;
- Geração dinâmica da página principal contendo a lista de recomendação na forma de uma div HTML. Para tal, com base no handle, o sistema deverá recuperar o título e os autores do recurso.

A justificativa da recomendação por popularidade é que, em um primeiro acesso ao repositório, o sistema não tem ainda nenhum indício do que o usuário procura, então a filtragem colaborativa seria mais adequada, pois veria o que os outros usuários já acessaram. Quando o usuário entra na página de um recurso, a recomendação por conteúdo é mais adequada, pois iria recomendar itens parecidos com o que o usuário acessa naquele momento.

O Diagrama de Sequência da Figura 15 apresenta o acesso à página principal do DSpace por parte de um usuário autenticado. O método de recomendação implementado deve realizar as seguintes ações:

- Solicitar a recomendação por Filtragem Colaborativa ou Baseada em Conteúdo (de escolha do desenvolvedor) ao RecOnt, utilizando o WebService RecOnt. Conforme previsto na API, esta chamada deverá conter o identificador do usuário e o identificador do tipo de recomendação solicitada;
- Receber a lista de recursos recomendados na forma de uma estrutura JSON. Cada recurso será identificado pelo seu handle;

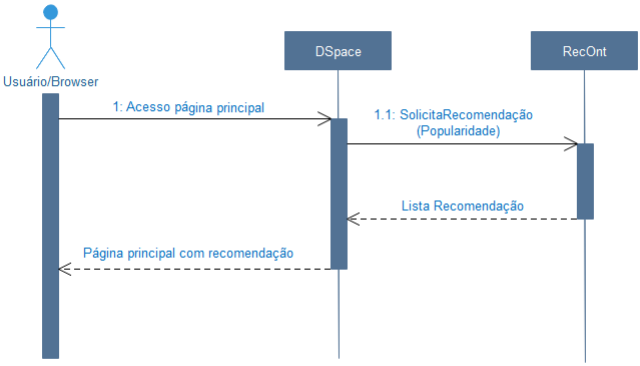

Figura 14 – Diagrama de sequência de um registro de um recurso baseado em popularidade

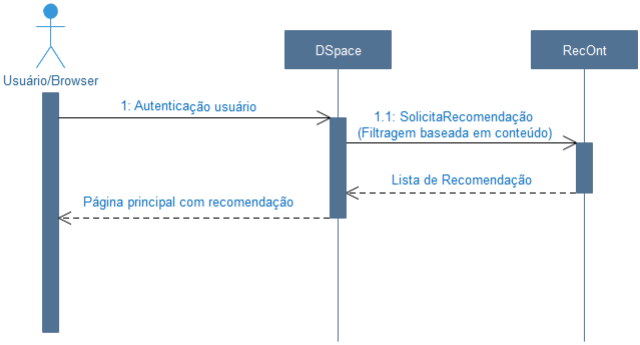

Figura 15 – Diagrama de sequência de um acesso de um recurso baseado em conteúdo

- Geração dinâmica da página principal contendo a lista de recomendação na forma de uma div HTML. Para tal, com base no

handle, o sistema deverá recuperar o título e os autores do recurso.

O método da RecOnt para solicitação de recomendação deve seguir o seguinte formato:

- **Método GET**:

  Exemplo: `http://localhost:8080/{ApplicationName}/{WebService}/rec/apiKey/{user}/nresult/rectype`

  Onde temos os seguintes parâmetros para a construção da estrutura da URL como no exemplo acima:

  - ApplicationName: [alfanumérico] Nome da aplicação;
  - WebService: [alfanumérico] Classe contendo os métodos do Web service;
  - rec: Método que será executado.
  - apiKey: [alfanumérico] Token responsável pela identificação do repositório cliente;
  - user: [alfanumérico] Usuário alvo da recomendação;
  - nresult: [inteiro] Numero de resultados;
  - recType: [inteiro] Tipo da recomendação (1: Por popularidade, 2: Baseada em conteúdo, 3: Colaborativa).

4.4 IMPLEMENTAÇÃO

Antes de mais nada, é necessário deixar claro que foi imprescindível o uso de um ambiente de desenvolvimento integrado (IDE) para o desenvolvimento da proposta. Para isto, escolheu-se o IntelliJ IDEA da empresa JetBrains (`https://www.jetbrains.com/idea/`). Sua versão completa disponibiliza uma interface bem adaptada para *debugs* on-the-fly em aplicações como o Tomcat.

O sistema implementado utiliza-se de duas classes Java inseridas no aspecto *viewArtifacts* localizado dentro do módulo dspace-xmlui:

- *AccessRegister*: Responsável pela coleta dos metadados do recurso, construção da ontologia e envio das informações via HTTP;

- *RecOnt*: Encarrega-se da solicitação dos dados no formato JSON a partir do servidor de recomendação e trata-os para a construção de recursos dentro do DSpace.

4.4.1 AccessRegister

A classe AccessRegister contém um método principal chamado *callWebService* que se encarrega de abrir uma conexão HTTP e construir a ontologia passando os metadados como variáveis, conforme mostrada na figura 16. Essa construção é atribuída à uma *string* onde faz-se sucessivas concatenações entre o texto imutável e as variáveis coletadas quando o usuário acessa o item. Essas variáveis são coletadas através de métodos *getters* a partir de uma instanciação da classe AccessRegister dentro da classe nativa do DSpace *ItemViewer*, responsável por controlar a exibição de um item no DSpace. A assinatura do método recebe como parâmetros as *string* dos metadados coletados.

```
{
   public void callWebService(String author, String subject,
        String handle, String eperson)
}
```

Figura 16 – Ontologia resultante com os metadados concatenados.

O método *getAuthor*, que pode ser observado abaixo, utiliza-se da classe *DSpaceObject* assim como o *getSubject* que contém a mesma lógica em seu código. No método *getEperson*, criado para retornar o usuário atual, utiliza-se a classe *Context* onde se está localizado o usuário enquanto que *getHandle* faz-se o uso das duas classes para o retorno do *Handle* do item no DSpace.

Abaixo, são mostradas essas três variações:

```
{
   public String getAuthor(DSpaceObject dso) throws SQLException
        {

        List<MetadataValue> author =
            itemService.getMetadataByMetadataString((Item)
            dso,"dc.contributor.author");
        this.author = author.get(0).getValue();
        this.author = this.author.replaceAll("\\s",""); //Remove
            espacos em brancos o qual a ontologia no aceita.
        return this.author;
    }
}
```

Neste método podemos observar que é criada uma lista para armazenar a *string* autor através do método *getMetadataByMetadataString* da classe *ItemService*. Em seguinte executa-se uma ação de remoção de espaços em branco, pois a ontologia não funcionará se existir espaços em branco nas variáveis.

```
{
   public String getEperson(Context context) throws SQLException
        {
        EPerson eperson = context.getCurrentUser();
        this.eperson = eperson.getID().toString();
        return this.eperson;
    }
}
```

Para coletar o usuário, é necessário o uso da classe *Context*, pois é só nela que é passado o usuário dentro do sistema. A classe *DSpaceObject* é construída para os metadados de um item além de outras informações não relevantes para este trabalho.

```
{
   public String getHandle(DSpaceObject dso, Context context)
       throws SQLException {
       ArrayList<String> identifiers =
           itemService.getIdentifiers(context, (Item) dso);
       this.handle = identifiers.get(0);
       this.handle =
           this.handle.substring(this.handle.lastIndexOf("/")+1);
       return this.handle;
    }
}
```

O *getHandle* em seu corpo, opera com o método *getIdentifiers* da classe *ItemService*. Para isso, usa-se tanto o *Context* quanto o *DSpaceObject* como seus parâmetros.

Na parte do envio do acesso, é criado a requisição HTTP pelo método POST com a construção do cabeçalho pelo código

```
{
       connection.setRequestMethod("POST");
       connection.setRequestProperty("User-Agent", "Mozilla/5.0
           (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
           Gecko) Ubuntu Chromium/53.0.2785.143
           Chrome/53.0.2785.143 Safari/537.36");
       connection.setRequestProperty("Accept-Language",
           "pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4");
       connection.setRequestProperty("Content-Type",
           "application/xml");
   }
}
```

e enviado através da execução do trecho

```
{
       // Envio da requisicao POST
       connection.setDoOutput(true);
       DataOutputStream wr = new
           DataOutputStream(connection.getOutputStream());
       wr.writeBytes(ontology);
       wr.flush();
       wr.close();

       BufferedReader in = new BufferedReader(
```

```
            new InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }
        in.close();
}
```

4.4.2 RecOnt

A classe *RecOnt*, por ser uma classe de exibição, faz o uso dos métodos *addBody* e *addPageMeta* já mencionados e estende a classe *AccessRegister*. Dessa forma, é montado, via API, uma divisão na página inicial responsável por listar itens que servirão para a recomendação. Dessa maneira, recupera-se os dados no formato JSON e fazendo o uso da biblioteca Gson, trata-se a string de retorno. Dessa string, é extraído o Handle que servirá como parâmetro para instanciar um objeto do DSpace pela classe *DSpaceObject* onde este representará um item na lista. Para repetir o processo, faz-se a utilização de um laço extraindo quantos Handles estiverem retornados na consulta ao servidor, criando assim uma lista de itens recomendados na página inicial.

O código tem duas seções que se comportam de forma bastante parecidas. Em um primeiro momento, verifica-se se o usuários está autenticado com o código

```
{
        if (context.getCurrentUser() != null) {
            String epersonId = null;
            epersonId =
                 context.getCurrentUser().getID().toString();
}
```

Se o usuário estiver autenticado, é executado o trecho para a apresentação de itens por conteúdo. Caso contrário, por popularidade.

Um ponto importante do código é a lógica para a construção da lista dos itens de recomendação como mostrado abaixo.

```
{
                Division home = body.addDivision("site-home",
                    "primary repository");
                Division recomendation =
                    home.addDivision("site-recomendation",
                    "secondary recomendation");
                recomendation.setHead("Itens sugeridos");

                JsonObject obj = root.getAsJsonObject()
                                     .getAsJsonObject("recommendation");
                obj = obj.getAsJsonObject("entry");
                String handle = "";
                for (int i = 0; i < obj.entrySet().size() - 1;
                    i++) {
                   handle = obj.get("value").getAsString();

                   DSpaceObject dso =
                       handleService.resolveToObject(context,
                       "123456789/" + handle);

                   ReferenceSet recomendationSet =
                       recomendation.addReferenceSet(
                          "site-recomendation",
                              ReferenceSet.TYPE_SUMMARY_LIST,
                          null, "recomendation");
                   recomendationSet.addReference(dso);
}
```

Como observa-se, a partir do JSON enviado pelo sistema de recomendação, cria-se um laço onde para cada item instanciado é adicionado em um conjunto de recomendações. A partir disso, o DSpace se encarrega de criar a seção de "Itens Sugeridos"na página inicial.

Nos exemplos a seguir, são mostrados os retornos do sistema de recomendação na chamada da classe *RecOnt*:

Exemplo JSON de retorno do Handle a partir de recomendação baseada em conteúdo:


```
{
   "audience":"38efcc71-46cd-4b62-9dbc-6f267f99d427",
   "recommendationType":"2",
   "recommendation":{
      "entry":{
         "key":"1",
```

```
        "value":"4"
      }
   }
}
```

Neste exemplo, 38efcc71-46cd-4b62-9dbc-6f267f99d427 representa a ID do usuário no DSpace. O tipo de recomendação é 2, portanto é um recomendação baseada em conteúdo, e seu Handle igual a 4.

Como exemplo de popularidade temos o seguinte retorno:

```
{
   "audience":"anonymous",
   "recommendationType":"1",
   "recommendation":{
      "entry":[
         {
            "key":"1",
            "value":"Resource"
         },
         {
            "key":"2",
            "value":"9"
         },
         {
            "key":"3",
            "value":"6"
         },
         {
            "key":"4",
            "value":"4"
         }
      ]
   }
}
```

Aqui, os Handles retornados são o 9, 6 e 4, formando uma lista de itens de recomendação por popularidade.

Após passar os Handles como parâmetros para instanciar os itens do DSpace, temos o resultado apresentado na figura 17.

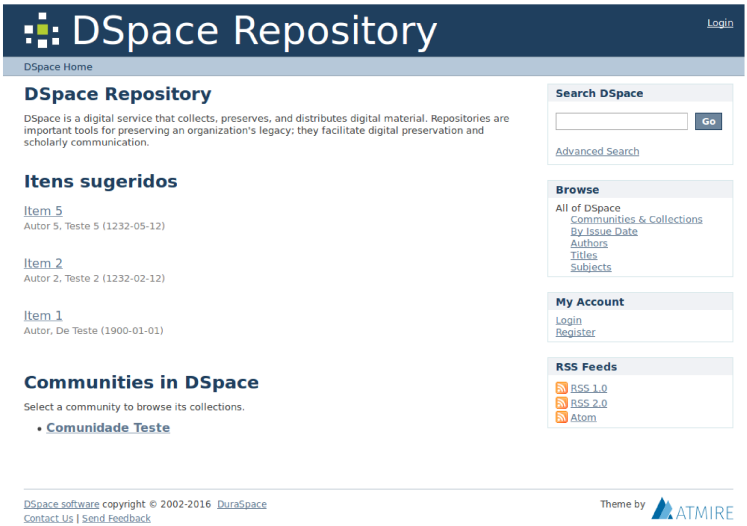

Figura 17 – Lista de recomendações por popularidade.

4.4.3 Sitemap

Por último, para o DSpace de fato criar a interface na posição da página desejada, é adicionado no arquivo *sitemap.xmap* as classes para fazer o mapeamento pelo Apache Cocoon, como mostrado nas marcações abaixo.

Declaração das classes no sitemap:

```
<map:transformer name="AccessRegister"
    src="org.dspace.app.xmlui.aspect.viewArtifacts.AccessRegister"/>
<map:transformer name="RecOnt"
    src="org.dspace.app.xmlui.aspect.viewArtifacts.RecOnt"/>
```

Inclusão da classe RecOnt na página inicial:

```
<!-- Display the DSpace homepage. This includes the news.xml
    file along with a list of top level communities in DSpace.
    -->
<map:match pattern="">
```

```
    <map:transform type="Include"
        src="resource://aspects/ViewArtifacts/dspace-home.xml" />
    <!-- DSpacePropertyFileReader will read the DSpace property
        file and place the selected properties value in this
        scope
    -->
        <map:act type="DSpacePropertyFileReader">
               <map:parameter name="dspace.dir"
                   value="dspace.dir" />
               <map:transform type="Include"
                   src="file://{dspace.dir}/config/news-xmlui.xml"
                   />
        </map:act>

      <map:transform type="RecOnt"/> <!-- Classe RecOnt na
          pagina inicial-->
        <map:serialize type="xml"/>
</map:match>
```

Inclusão da classe AccessRegister na página do item:

```
<map:select type="browser">
<map:when test="spider">
   <map:select type="IfModifiedSinceSelector">
      <map:when test="true">
         <map:act type="NotModifiedAction" />
         <map:serialize />
      </map:when>
      <map:otherwise>
         <map:transform type="ItemViewer" />
         <map:transform type="AccessRegister"/> <!-- Classe
             AccessRegister na pagina pertencente ao item -->
         <map:serialize type="xml" />
      </map:otherwise>
   </map:select>
</map:when>
<map:otherwise>
   <map:transform type="ItemViewer" />
   <map:transform type="AccessRegister"/> <!-- Classe
       AccessRegister na pagina pertencente ao item -->
   <map:serialize type="xml" />
</map:otherwise>
</map:select>
```

4.5 AVALIAÇÃO DA INTEGRAÇÃO DO SISTEMA DE RECOMENDAÇÃO E TESTES

O objetivo deste trabalho não é o de avaliação de uma técnica específica de recomendação, a ser implantada no servidor RecOnt. O propósito deste trabalho é avaliar o impacto na adaptação do repositório DSPace para atuar como um cliente do serviço de recomendação.

Houve uma certa dificuldade em termos da própria inclusão de funcionalidades no DSpace. Mas após a identificação dos passos para esta customização, a inclusão das funções de registro de acesso e apresentação da recomendação foram bastante facilitadas. Neste sentido, considera-se que o uso de um Web service para acesso ao sistema de recomendação resultou efetivamente na redução da complexidade de sua integração com o repositório DSpace.

Em termos de testes, foram realizados medidas de tempo de resposta, para avaliar os atrasos de registro de acesso a recursos, e de apresentação da recomendação. No cenário de teste, tanto o lado cliente, como o servidor de recomendação estavam localizada na rede UFSC, sendo que o atraso de ida-e-volta de rede é inferior a 1ms.

Para o registro de acessos, foram simulados 4500 acessos a recursos. O atraso médio de registro destes acessos foi de 1,34s (com desvio padrão de 0,23) apresentados no gráfico da figura 18. Dado que o registro de acesso é assíncrono, este atraso não compromete a usabilidade do repositório. Mesmo assim, como este tempo é inferior a 2s, tempos limites de atrasos convencional da Web, conforme estudo feito por Nah (2004), o resultado considera-se satisfatório. Conforme análise durante os testes, o componente que resultou em um maior uso de recursos de CPU e memória foi o próprio Tomcat. O Neo4J limitou-se a 1% de uso de CPU. Possíveis atualizações do Tomcat, e de configuração para melhor desempenho, poderão melhorar estes tempos, visto que a configuração utilizada foi a padrão.

Em termos de tempo de resposta da apresentação, foram simuladas 100 recomendações. Na recomendação por popularidade, o tempo médio de resposta foi de 0,64s (Desvio Padrão de 0,13) como apresentado no gráfico da figura 19. Já na recomendação por conteúdo, foi realizada inicialmente a simulação de 5 recursos por parte do usuário, e em seguida foram realizadas as medidas do tempo de resposta de 100 recomendações. Neste caso, o atraso médio foi de 0,63s (Desvio Padrão de 0,12) como visto no gráfico da figura 20.

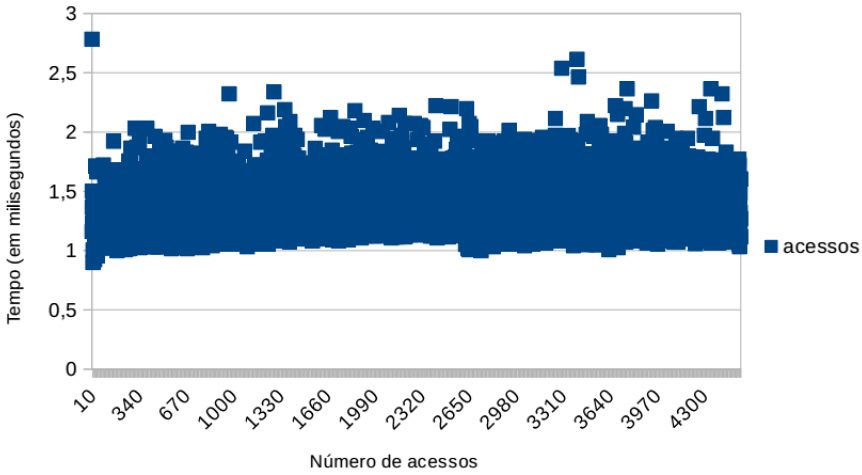

Figura 18 – Tempo de registro de acesso ao recurso no sistema de recomendação.

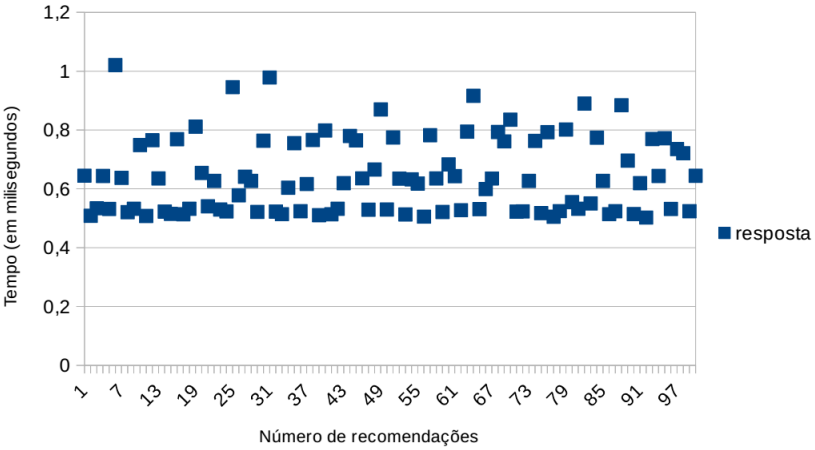

Figura 19 – Tempo de resposta da apresentação baseado em recomendação por popularidade.

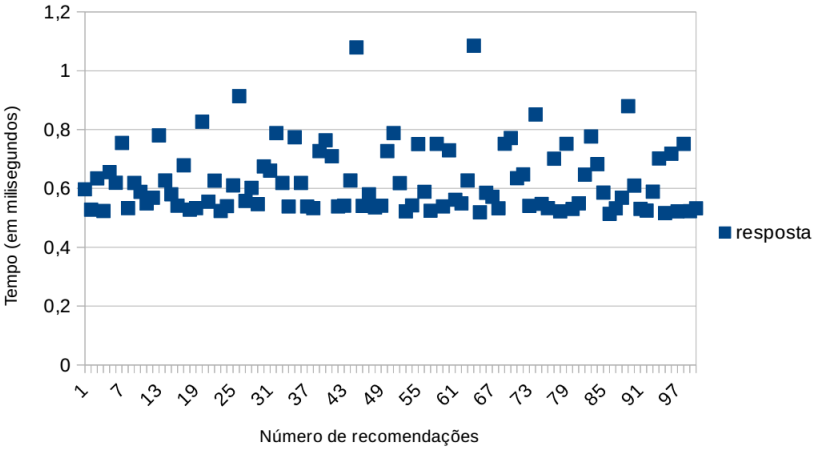

Figura 20 – Tempo de resposta da apresentação baseado em recomendação baseada em conteúdo.

Normalmente, devido à complexidade do processo da recomendação, o atraso da recomendação por conteúdo deveria ser maior que o tempo por popularidade. No caso dos experimentos, como o histórico de acessos do usuário foi pequeno, os tempos de repostas dos tipos de recomendação foram similares.

5 CONCLUSÃO

Até hoje não há solução disponível de sistema de recomendação para DSpace e sua implementação se torna importante visto que mostra-se ser uma grande aliado na localização rápida de conteúdos do interesse do usuário. Mas ao mesmo tempo, uma solução para estes sistemas não pode ser de difícil integração, haja visto que maior parte das organizações não dispõem de equipe especializada no DSpace.

Este trabalhou ocupou-se em estudar meios de uma implementação efetiva de um cliente Web service dentro da plataforma para repositórios digitais DSpace, além de apresentar conceitos e definições responsáveis pela fundamentação teórica do referente trabalho. O sistema de recomendação proposto foi implantando com sucesso na plataforma DSpace cumprindo seu objetivo de ser fracamente acoplável e adaptável ao domínio dos conteúdos de um RD, fato este que justifica pelo seu padrão Web service de projeto.

Apesar de relativamente simples, para se chegar ao estado final deste presente trabalho, foi realizado um grandioso esforço de reconhecimento do código-fonte e funcionamento do DSpace como um todo. Esforço que foi necessário devido a sua documentação escassa e não clara juntamente com uma comunidade de desenvolvedores ainda escondida no que se trata de de personalização e customização do código-fonte. O código-fonte é pouco documentado mostrando-se útil somente para quem fez parte de seu projeto de implementação. O fato do DSpace possuir muitos softwares de terceiros amarrados em si para seu funcionamento dificulta ainda mais para manter o sistema estável antes e após a alteração no código-fonte.

5.1 TRABALHOS FUTUROS

Para alguns dos trabalhos futuros, podemos citar:

- Teste do sistema de recomendação com itens e usuários reais;
- Otimização e correção de eventuais bugs do código-fonte criado;
- Implementação do sistema de recomendação no sistema em produção;

- Mudança da exibição da lista de recomendação criada para recomendações baseadas em conteúdo da página inicial para a página de recurso.

REFERÊNCIAS

ANJORIN, M. et al. A framework for cross-platform graph-based recommendations for tel. In: *Proceedings of the 2nd workshop on recommender systems in technology enhanced learning*. [S.l.: s.n.], 2012. p. 83–88.

ARELLANO, M. A. Preservação de documentos digitais. *Ci. Inf., Brasília*, SciELO Brasil, v. 33, n. 2, p. 15–27, 2004.

BEHAM, G. et al. Recommending knowledgeable people in a work-integrated learning system. *Procedia Computer Science*, Elsevier, v. 1, n. 2, p. 2783–2792, 2010.

CASAGRANDE, M. F. R. Técnica de recomendação para repositórios digitais baseada em metadados e agrupamento de usuários. 2014.

CASTAGNé, M. *Institutional repository software comparison: DSpace, EPrints, Digital Commons, Islandora and Hydra*. Aug 2013. (GSS cIRcle Open Scholar Award (UBCV Non-Thesis Graduate Work)). <https://open.library.ubc.ca/cIRcle/collections/42591/items/1.0075768>.

DCMI, D. C. M. I. et al. *Dublin core metadata element set, version 1.1*. [S.l.]: Dublin Core Metadata Initiative, 2012.

DSPACE. *Dspace - dspace is a turnkey institutional repository application*. 2016. <http://www.dspace.org/>.

DSPACE, D. T. *DSpace 5.x Documentation*. 2016. <https://wiki.duraspace.org/display/DSDOC5x>.

DURASPACE, C. *XMLUI Configuration and Customization*. <https://wiki.duraspace.org/display/DSDOC5x/XMLUI+Configuration+and+Customization>.

ELLIOTT, D.; RUTHERFORD, J.; ERICKSON, J. *A recommender system for the DSpace open repository platform*. [S.l.], 2008.

GROUP, W. O. W. *Owl 2 web ontology language document overview (second edition)*. 2012. <http://www.w3.org/TR/owl2-overview/>.

HEERY, R.; ANDERSON, S. Digital repositories review. Joint Information Systems Committee, 2005.

IBICT, I. B. d. I. e. C. e. T. *Sobre Repositórios Digitais*. 2016. <http://www.ibict.br/informacao-para-ciencia-tecnologia-e-inovacao/repositorios-digitais>.

LEMOS, F. D. et al. Improving photo recommendation with context awareness. In: ACM. *Proceedings of the 18th Brazilian symposium on Multimedia and the web*. [S.l.], 2012. p. 321–330.

NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, Taylor & Francis, v. 23, n. 3, p. 153–163, 2004.

OLIVEIRA, R. R. de; CARVALHO, C. L. de. Bibliotecas digitais e o repositório fedora. 2011.

PHILLIPS, S. et al. Manakin. 2005.

RICCI, F. et al. *Introduction to recommender systems handbook*. [S.l.]: Springer, 2011.

SALLES, A.; WILLRICH, R. Recommending web service based on ontologies for digital repositories. In: ACM. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.], 2015. p. 65–72.

SHINTAKU, M.; MEIRELLES, R. F. Manual do dspace: administração de repositórios. EDUFBA, 2010.

SMITH, M. et al. Dspace: An open source dynamic digital repository. Corporation for National Research Initiatives, 2003.

TANSLEY, R. et al. The dspace institutional digital repository system: current functionality. In: IEEE COMPUTER SOCIETY. *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*. [S.l.], 2003. p. 87–97.

WOLF, A. S.; MONTEIRO, A. P. L.; VALMORBIDA, W. Biblioteca digital da univates utilizando o software dspace. *Destaques Acadêmicos*, v. 1, n. 4, 2013.

ANEXO A – AccessRegister

```
package org.dspace.app.xmlui.aspect.viewArtifacts;

import org.dspace.app.xmlui.cocoon.AbstractDSpaceTransformer;
import org.dspace.app.xmlui.wing.WingException;
import org.dspace.authorize.AuthorizeException;
import org.dspace.content.DSpaceObject;
import org.dspace.content.Item;
import org.dspace.content.MetadataValue;
import org.dspace.content.service.ItemService;
import org.dspace.core.Context;
import org.dspace.eperson.EPerson;
import org.xml.sax.SAXException;
import org.dspace.content.factory.ContentServiceFactory;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 * Created by bernardo on 01/04/16.
 */
public class AccessRegister extends AbstractDSpaceTransformer {

    String handle;
    String author;
    String title;
    String subject;
    String eperson;
    ItemService itemService =
        ContentServiceFactory.getInstance().getItemService();

    public AccessRegister() throws SQLException {
    }

    public String getAuthor(DSpaceObject dso) throws
        SQLException {
```

```
        List<MetadataValue> author =
            itemService.getMetadataByMetadataString((Item)
            dso,"dc.contributor.author");
        this.author = author.get(0).getValue();
        this.author = this.author.replaceAll("\\s",""); //Remove
            espacos em brancos o qual a ontologia no aceita.
        return this.author;
    }

    public String getTitle(DSpaceObject dso) throws SQLException
        {

        List<MetadataValue> title =
            itemService.getMetadataByMetadataString((Item)
            dso,"dc.title");
        this.title = title.get(0).getValue();
        return this.title;
    }

    public String getSubject(DSpaceObject dso) throws
        SQLException {

        List<MetadataValue> subject =
            itemService.getMetadataByMetadataString((Item)
            dso,"dc.subject");
        this.subject = subject.get(0).getValue();
        this.subject = this.subject.replaceAll("\\s","");
            //Remove espacos em brancos o qual a ontologia no
            aceita.
        return this.subject;
    }

    public String getEperson(Context context) throws
        SQLException {
        EPerson eperson = context.getCurrentUser();
        this.eperson = eperson.getID().toString();
        return this.eperson;
    }

    public String getHandle(DSpaceObject dso, Context context)
        throws SQLException {
        ArrayList<String> identifiers =
            itemService.getIdentifiers(context, (Item) dso);
        this.handle = identifiers.get(0);
```

```
        this.handle =
            this.handle.substring(this.handle.lastIndexOf("/")+1);
        return this.handle;
    }

    public void callWebService(String author, String subject,
         String handle, String eperson)
            throws SAXException, WingException,
                AuthorizeException, IOException, SQLException {

        // Mtodo  POST
        URL url = new URL ("http://{host}:8080/WebServiceProject
                      /wsTeste/access/DSpaceTest/Context/" +
                          System.currentTimeMillis());
        HttpURLConnection connection =
            (HttpURLConnection)url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("User-Agent", "Mozilla/5.0
            (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
            Gecko) Ubuntu Chromium/53.0.2785.143
            Chrome/53.0.2785.143 Safari/537.36");
        connection.setRequestProperty("Accept-Language",
            "pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4");
        connection.setRequestProperty("Content-Type",
            "application/xml");

        //Ontologia
        String ontology = "<?xml version=\"1.0\"?>\n" +
               "<!DOCTYPE Ontology [\n" +
               "\n" +
               "\n" +
               "<!ENTITY xsd
                   \"http://www.w3.org/2001/XMLSchema#\" >\n" +
               "<!ENTITY xml
                   \"http://www.w3.org/XML/1998/namespace\" >\n"
                   +
               "<!ENTITY rdfs
                   \"http://www.w3.org/2000/01/rdf-schema#\"
                   >\n" +
               "<!ENTITY rdf
                   \"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"
                   >]>\n" +
               "<Ontology\n" +
               "\txmlns=\"http://www.w3.org/2002/07/owl#\"\n" +
```

```
"\txml:base=\"http://repositorio.ufsc.br/access\"\n"
    +
"\txmlns:rdf=\"http://www.w3.org/
                  1999/02/22-rdf-syntax-ns#\"\n" +
"\txmlns:xml=\"http://www.w3.org/XML/1998/namespace\"\n"
    +
"\txmlns:xsd=\"http://www.w3.org/2001/XMLSchema#\"\n"
    +
"\txmlns:rdfs=\"http://www.w3.org/2000/01/rdf-schema#\"\n"
    +
"\tontologyIRI=\"http://repositorio.ufsc.br/access\">\n"
    +
"\t<Prefix name=\"\"
    IRI=\"http://repositorio.ufsc.br/access#\"/>\n"
    +
"\t<Prefix name=\"owl\"
    IRI=\"http://www.w3.org/2002/07/owl#\"/>\n" +
"\t<Prefix name=\"rdf\"
           IRI=\"http://www.w3.org/
                     1999/02/22-rdf-syntax-ns#\"/>\n"
                         +
"\t<Prefix name=\"xsd\"
    IRI=\"http://www.w3.org/2001/XMLSchema#\"/>\n"
    +
"\t<Prefix name=\"rdfs\"
    IRI=\"http://www.w3.org/2000/01/rdf-schema#\"/>\n"
    +
"\t<Prefix name=\"RecOnt\"
    IRI=\"http://{pathToOWL}/RecOnt.owl#\"/>\n" +
"\t<Prefix name=\"DCContext\"
    IRI=\"http://{pathToOWL}/DCContext.owl#\"/>\n"
    +
"\t<Import>http://{pathToOWL}/DCContext.owl</Import>\n"
    +
"\t<Declaration>\n" +
"\t\t<NamedIndividual IRI=\"#"+author+"\"/>\n" +
"\t</Declaration>\n" +
"\t<Declaration>\n" +
"\t\t<NamedIndividual IRI=\"#"+subject+"\"/>\n" +
"\t</Declaration>\n" +
"\t<Declaration>\n" +
"\t\t<NamedIndividual IRI=\"#"+eperson+"\"/>\n" +
"\t</Declaration>\n" +
"\t<Declaration>\n" +
"\t\t<NamedIndividual IRI=\"#"+handle+"\"/>\n" +
```

```
            "\t</Declaration>\n" +
            "\t<ClassAssertion>\n" +
            "\t\t<Class
                abbreviatedIRI=\"DCContext:Author\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+author+"\"/>\n" +
            "\t</ClassAssertion>\n" +
            "\t<ClassAssertion>\n" +
            "\t\t<Class
                abbreviatedIRI=\"DCContext:Subject\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+subject+"\"/>\n" +
            "\t</ClassAssertion>\n" +
            "\t<ClassAssertion>\n" +
            "\t\t<Class
                abbreviatedIRI=\"RecOnt:Audience\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+eperson+"\"/>\n" +
            "\t</ClassAssertion>\n" +
            "\t<ClassAssertion>\n" +
            "\t\t<Class
                abbreviatedIRI=\"DCContext:Resource\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+handle+"\"/>\n" +
            "\t</ClassAssertion>\n" +
            "\t<ObjectPropertyAssertion>\n" +
            "\t\t<ObjectProperty
                abbreviatedIRI=\"RecOnt:acessed\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+eperson+"\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+handle+"\"/>\n" +
            "\t</ObjectPropertyAssertion>\n" +
            "\t<ObjectPropertyAssertion>\n" +
            "\t\t<ObjectProperty
                abbreviatedIRI=\"DCContext:hasAuthor\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+handle+"\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+author+"\"/>\n" +
            "\t</ObjectPropertyAssertion>\n" +
            "\t<ObjectPropertyAssertion>\n" +
            "\t\t<ObjectProperty
                abbreviatedIRI=\"DCContext:hasSubject\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+handle+"\"/>\n" +
            "\t\t<NamedIndividual IRI=\"#"+subject+"\"/>\n" +
            "\t</ObjectPropertyAssertion>\n" +
            "</Ontology>";

        // Envio da requisicao POST
        connection.setDoOutput(true);
        DataOutputStream wr = new
            DataOutputStream(connection.getOutputStream());
```

```
        wr.writeBytes(ontology);
        wr.flush();
        wr.close();

        BufferedReader in = new BufferedReader(
                new
                    InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();
    }
}
```

ANEXO B – RecOnt

```
package org.dspace.app.xmlui.aspect.viewArtifacts;

import com.google.gson.*;
import org.dspace.app.xmlui.wing.WingException;
import org.dspace.app.xmlui.wing.element.Body;
import org.dspace.app.xmlui.wing.element.Division;
import org.dspace.app.xmlui.wing.element.PageMeta;
import org.dspace.app.xmlui.wing.element.ReferenceSet;
import org.dspace.authorize.AuthorizeException;
import org.dspace.content.DSpaceObject;
import org.dspace.handle.factory.HandleServiceFactory;
import org.dspace.handle.service.HandleService;
import org.xml.sax.SAXException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.sql.SQLException;

/**
 * Created by bernardo on 02/09/16.
 */
public class RecOnt extends AccessRegister {

    public HandleService handleService
            =
                HandleServiceFactory.getInstance().getHandleService();

    public RecOnt() throws SQLException {
        super();
    }

    public void addPageMeta(PageMeta pageMeta) throws
         SAXException,
            WingException, AuthorizeException
    {
        pageMeta.addMetadata("title").addContent("About Us");
    }

    @SuppressWarnings("Duplicates")
    public void addBody(Body body) throws SAXException,
         WingException, AuthorizeException, IOException,
```

```
 SQLException {

//  Recomendao  por  contedo
if (context.getCurrentUser() != null) {
    String epersonId = null;
    epersonId =
        context.getCurrentUser().getID().toString();

    URL url = new
        URL("http://{host}:8080/WebServiceProject/
                    wsTeste/rec/DSpaceTest/" +
                        epersonId + "/3/2");
    HttpURLConnection connection = (HttpURLConnection)
        url.openConnection();
    connection.setRequestMethod("GET");
    connection.connect();
    BufferedReader bufferReader = new BufferedReader(new
        InputStreamReader(connection.getInputStream()));
    String str;
    StringBuffer stringBuffer = new StringBuffer();
    while ((str = bufferReader.readLine()) != null) {
        stringBuffer.append(str);
        stringBuffer.append("\n");
    }
    String jsonString = stringBuffer.toString();

    JsonElement root = new JsonParser().parse(jsonString);
    Object[] recomendationValue =
        root.getAsJsonObject().entrySet().toArray();
    //Verifica se h item para recomendar. Caso no, a
        lista e a montagem da seo no  so  formadas
    if(!(recomendationValue[2].toString()
            .equals("recommendation=null"))){

        Division home = body.addDivision("site-home",
            "primary repository");
        Division recomendation =
            home.addDivision("site-recomendation",
            "secondary recomendation");
        recomendation.setHead("Itens sugeridos");

        JsonObject obj =
            root.getAsJsonObject().getAsJsonObject("recommendation");
        obj = obj.getAsJsonObject("entry");
        String handle = "";
```

```
            for (int i = 0; i < obj.entrySet().size() - 1;
                 i++) {
                handle = obj.get("value").getAsString();

                DSpaceObject dso =
                    handleService.resolveToObject(context,
                    "123456789/" + handle);

                ReferenceSet recomendationSet =
                    recomendation.addReferenceSet(
                       "site-recomendation",
                           ReferenceSet.TYPE_SUMMARY_LIST,
                       null, "recomendation");
                recomendationSet.addReference(dso);
            }
        }
    }
    //  Recomendao  por popularidade
    else{
        URL url = new
            URL("http://{host}:8080/WebServiceProject/
                        wsTeste/rec/DSpaceTest/user/3/1");
        HttpURLConnection connection = (HttpURLConnection)
            url.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();
        BufferedReader bufferReader = new BufferedReader(new
            InputStreamReader(connection.getInputStream()));
        String str;
        StringBuffer stringBuffer = new StringBuffer();
        while ((str = bufferReader.readLine()) != null) {
            stringBuffer.append(str);
            stringBuffer.append("\n");
        }
        String jsonString = stringBuffer.toString();

        JsonElement root = new JsonParser().parse(jsonString);
        Object[] recomendationValue =
            root.getAsJsonObject().entrySet().toArray();
        //Verifica se h item para recomendar. Caso no, a
            lista e a montagem da seo  no  so  formadas
        if(!(recomendationValue[2].toString()
                              .equals("recommendation=null")))
                                  {
```

```
                JsonObject obj = root.getAsJsonObject()
                                  .getAsJsonObject("recommendation");


                Division home = body.addDivision("site-home",
                    "primary repository");
                Division recomendation =
                    home.addDivision("site-recomendation",
                    "secondary recomendation");
                recomendation.setHead("Itens sugeridos");

                JsonArray array = obj.getAsJsonArray("entry");
                String handle;
                JsonObject obj2;
                for (int i = 1; i < array.size(); i++) {
                    obj2 = (JsonObject) array.get(i);
                    handle = obj2.get("value").getAsString();

                    DSpaceObject dso =
                        handleService.resolveToObject(context,
                        "123456789/" + handle);

                    ReferenceSet recomendationSet =
                        recomendation.addReferenceSet(
                           "site-recomendation",
                               ReferenceSet.TYPE_SUMMARY_LIST,
                           null, "recomendation");
                    recomendationSet.addReference(dso);
                }
            }
        }
    }
}
```

ANEXO C – Artigo

Sistema de recomendação para a plataforma DSpace

Bernardo V. Engelke[1]

[1]Departamento de Informática e Estatística– Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brazil

bernardo.engelke@grad.ufsc.br

Resumo. *Repositórios Digitais estão ficando cada vez maiores e significativos dentro da Web, permitindo uma melhor distribuição do conhecimento produzido dentro de instituições ou empresas. Uma das soluções abertas mais adotadas para a implantação de Repositórios Digitais é o DSpace. Esta solução de repositório digital oferece diversas funcionalidades suportando o gerenciamento do fluxo de submissão de conteúdos, armazenamento e distribuição de diversos tipos de recursos digitais. Os tipos de recursos digitais dependem dos objetivos da organização ou pessoa, podendo ser trabalhos acadêmicos, como teses, dissertações, monografias e artigos, ou de caráter mais geral, como vídeos, atas e trabalhos artísticos. Com o crescente número de conteúdos digitais disponibilizados, fica cada vez mais complexa a tarefa dos usuários deste tipo de plataforma localizarem conteúdos de seu interesse. Uma forma usual de apoiar os usuários a localizar conteúdos de interesse é implantar Sistemas de Recomendação. Sistemas de Recomendação, integrados a Repositórios Digitais, aplicam técnicas de filtragem para indicar ao usuário uma lista de conteúdos do repositório com base no perfil deste usuário. Neste sentido, no presente projeto, propôs-se a implantação na plataforma DSpace de um sistema de recomendações utilizando um modelo Web service baseado em tecnologias da Web semântica. A escolha desta solução foi devido à facilidade de instalação destes serviço em repositórios digitais, com pouco impacto no código. A avaliação do impacto de instalação deste sistema de recomendação baseado em Web service no DSpace também é um dos objetivos deste projeto.*

1. Introdução

No contexto da Ciência da Informação, existe há muito tempo uma preocupação na preservação de perpetuação de documentos produzidos dentro e fora da esfera digital. Muito se tem pensado em modelos e soluções de armazenamento para que não haja perdas massivas de informações e que elas possam ser expostas livremente ao público. Foi com essa preocupação que surgiu a ideia da criação de plataformas que pudessem armazenar conteúdos digitais a fim de dar uma confiabilidade para a questão de segurança e preservação do armazenamento de mídias em formato digital[Arellano 2004].

O conceito de Repositórios Digitais (RDs) surgiu justamente para sanar essa necessidade de preservação e armazenamento a longo prazo. Conteúdos digitais podem ser disseminados de várias formas. Uma delas é através da criação de RDs, que armazenam coleções de conteúdos digitais, geralmente voltados a um escopo específico. Este tipo de sistema tem tido amplo crescimento ao longo dos últimos anos [Casagrande 2014].

Apesar de existirem soluções proprietárias, instituições geralmente tendem a usar soluções de código aberto para a implementação de seus RDs. Um expoente na classe desses softwares para a criação de RDs é o DSpace[DSpace 2016a].

A estrutura do DSpace é dividida em três principais categorias: Comunidades; Coleções; e Itens. Uma comunidade possui um conjunto de coleções que detém um conjunto de itens. Por sua vez, item contém um grupo de metadados representados pelo padrão Dublin Core (DC)[DCMI et al. 2012] sobre o conteúdo em questão, servindo para sua busca e referência dentro da plataforma[Tansley et al. 2003].

O DSpace é um software que visa operar como um serviço institucional centralizador, ou seja, diferentes comunidades inerentes à instituição, como laboratórios, departamentos e unidades administrativas podem ter suas próprias áreas dentro do sistema. Os usuários membros das comunidades podem então, a partir de uma interface Web, depositar seus materiais de interesse[Tansley et al. 2003].

Um exemplo de RD usando a plataforma DSpace é o Repositório Institucional da UFSC (`http://repositorio.ufsc.br`). O objetivo deste repositório é aumentar a visibilidade de sua produção acadêmica para dentro e fora de seu ambiente. Atualmente, o repositório da UFSC conta com mais de 80.000 itens dentro de seu banco de dados. Segundo a própria instituição, o Repositório Institucional (RI) possui como objetivos: Contribuir para o aumento da visibilidade dos pesquisadores e da produção científica da UFSC; preservar a memória intelectual da Universidade; reunir em um único local virtual e de forma permanente a produção científica e institucional, disponibilizando o livre acesso aos conteúdos digitais e ampliar e facilitar o acesso à produção científica de uma forma geral.[1]

Segundo Casagrande (2014), é crescente o número de RDs na forma de RIs. Consequentemente, cresce também o tamanho e números de coleções contidas nesses repositórios. Desse modo, devido a esse progressivo número de itens disponíveis, gera-se um problema de sobrecarga de informações fazendo com que o usuário realize um grande esforço para buscar um item de seu interesse.Uma forma de resolver essa questão é na implementação de um sistema de recomendações que registra o acesso do usuário a um item e cria-se uma lista de sugestões de itens com base nos metadados desejados pela implementação [Salles and Willrich 2015] pois a ferramentas utilizada pela instituição (DSpace) ainda não apresenta tal sistema.

Portanto, no presente trabalho, pretende-se utilizar de uma implementação via Web service como estratégia de solução, que, através desta, fornece-se um maior desacoplamento do sistema de recomendação com o RD, possibilitando inclusive a portabilidade para outras plataformas de conteúdo de preservação digital afins.

1.1. Objetivos

1.1.1. Objetivo Geral

O objetivo deste projeto de conclusão de curso é implantar na plataforma DSpace um Sistema de Recomendação. Para tal, é adotado um Web service REST de recomendação proposto por [Salles and Willrich 2015], que se apoia em tecnologias da Web Semântica

[1]Disponível em `https://repositorio.ufsc.br/` acessado em 8 de julho de 2016

e Banco de Dados baseados em grafos para gerar recomendação com base nos perfis dos usuários. Mais especificamente, o objetivo deste projeto é implementar um cliente deste Web service de recomendação integrado na plataforma DSpace. A meta é avaliar a complexidade de integração desta solução de sistema de recomendação na plataforma DSpace. Também serão realizados experimentos visando avaliar o sistema de recomendação proposto no contexto do Repositório Institucional da UFSC.

1.1.2. Objetivos Específicos

Além do objetivo principal, procura-se alcançar os seguintes objetivos específicos:

- Obter conhecimentos na área de sistemas de recomendação, e especificamente a solução de Web service de Recomendação proposto por [Salles and Willrich 2015];
- Implementar um cliente de Web service na plataforma DSpace que se conecte com um servidor onde situa-se a lógica de negócios do sistema de recomendação;
- Avaliar o impacto na adaptação do repositório DSpace para atuar como um cliente do serviço de recomendação.

1.2. Estrutura do trabalho

O trabalho está organizados da seguinte forma: Capítulo 1 procura introduzir os conceitos básicos que serão abordados neste trabalho, dando uma prévia de toda fundamentação

Figura 1. Estrutura hierárquica de comunidades, sub-comunidades e coleções do repositório institucional da Universidade Federal de Santa Catarina. Fonte: `https://repositorio.ufsc.br/`

teórica do trabalho. No capítulo 2 é apresentada a definição e soluções referentes à repositórios digitais. Em seguida no capítulo 3, aborda-se os sistemas de recomendação, apresentando a proposta para solução da problemática em questão. Por fim, o capítulo 4 trata da implementação e testes da solução proposta, finalizando com a conclusão apresentada no capítulo 5.

1.3. Repositórios Digitais

Este capítulo apresenta os principais conceitos relacionados a Repositórios digitais, incluindo definições, soluções abertas e aspectos técnicos.

1.4. Definição

Na análise de [Heery and Anderson 2005], os autores relatam uma preocupação em definir um repositório digital. Segundo eles, uma gama crescente de áreas de atividades dentro do contexto de ciência da informação referem-se a coleções de conteúdos depositadas como ”repositórios”. A fim de incentivar a comunicação entre essas áreas, e promover a interoperabilidade, é preciso definir as características de ”repositórios”e buscar a coerência de uma abordagem comum. Para tal, [Heery and Anderson 2005], propõem algumas dessas características para diferenciar repositórios digitais de qualquer outra coleção digital:

- O conteúdo precisa ser depositado em um repositório, quer seja pelo criador de conteúdo, proprietário ou terceiro;
- A arquitetura do repositório deve oferecer funcionalidades para gerenciar o conteúdo, bem como os metadados;
- O repositório deve oferecer um conjunto mínimo de serviços básicos, como, por exemplo, inserção, retorno, busca e controle de acesso;
- O repositório deve ser confiável, mantido e gerenciável.

Dentro do cenário brasileiro, O IBICT (Instituto Brasileiro de Informação em Ciência e Tecnologia) [IBICT 2016] define RDs como: Os repositórios digitais (RDs) são bases de dados online que reúnem de maneira organizada a produção científica de uma instituição ou área temática. Os RDs armazenam arquivos de diversos formatos. Ainda, resultam em uma série de benefícios tanto para os pesquisadores quanto às instituições ou sociedades científicas, proporcionam maior visibilidade aos resultados de pesquisas e possibilitam a preservação da memória científica de sua instituição. Os RDs podem ser institucionais ou temáticos. Os repositórios institucionais lidam com a produção científica de uma determinada instituição. Os repositórios temáticos com a produção científica de uma determinada área, sem limites institucionais.

1.5. Soluções abertas

Para a implementação de RDs como RIs, instituições são levadas a aderir a soluções de código aberto [Salles and Willrich 2015].

São exemplos de softwares de RDs: DSpace, EPrints, Digital Commons, OPUS e Fedora[de Oliveira and de Carvalho 2011, Castagné 2013].

O OpenDOAR (Diretório de Repositórios de Acesso Aberto em inglês) registra um uso majoritário (quase 50%) pelo software DSpace como observado na figura 2.

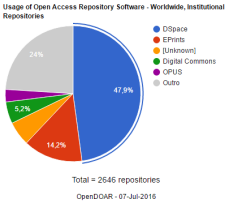

Figura 2. Estatística de uso de plataformas de RDs com código aberto. Fonte: `http://www.opendoar.org/find.php?format=charts`, acessado em 7 de Julho de 2016.

Levando em contas esses dados, foi escolhido o software DSpace para a realização da implementação da proposta. Portanto, nas próximas seções, o texto se concentrará em descrever com mais detalhes seu funcionamento e tecnologias usadas.

1.6. Dspace

Criado em 2002 pelo MIT e os pelos laboratórios da Hewlett Packard (HP), é um software para criação de RDs, em geral RIs, registrando um total de 1356 implementações da plataforma[2], tornando-o assim o software mais usado mundialmente para tal finalidade. É usado por bibliotecas, arquivos e centros de pesquisa[de Oliveira and de Carvalho 2011, Castagné 2013].

A plataforma DSpace possui uma estrutura baseada em comunidades e coleções, conseguindo assim, criar uma representação dos centros, departamento ou unidades administrativas de uma instituição, podendo abrigar os mais variados formatos de mídias digitais (figura 3)[de Oliveira and de Carvalho 2011].

Dentro de coleções são armazenados os itens que, juntamente com o conteúdo digital, possuem os metadados para sua classificação e indexação[Smith et al. 2003]. A figura 2 procura dar uma visão mais abrangente de todos os objetos que fazem parte da organização dentro do DSpace.

Segundo os autores do projeto, DSpace é um software gratuito de âmbito acadêmico, sem fins lucrativos e para construção de repositórios digitais abertos para organizações. A plataforma preserva e permite acesso fácil e livre para todos os tipos de conteúdos digitais, incluindo textos, imagens, imagens em movimento, mpegs e conjuntos de dados. Possui uma crescente comunidade de desenvolvedores que ajudam a expandir e melhorar ainda mais o software para cada versão lançada[DSpace 2016a].

O projeto ainda afirma que o DSpace é um software de fácil instalação, sendo ele completamente customizável para caber nas necessidades de qualquer organização.

[2]Dados retirados do ROAR em julho de 2013 (Registro de Repositórios de Acesso Aberto em inglês) - `http://roar.eprints.org/`

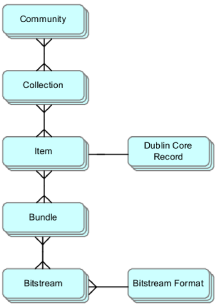

Figura 3. Visão geral da estrutura hierárquica dos objetos no DSpace. Fonte [Tansley et al. 2003]

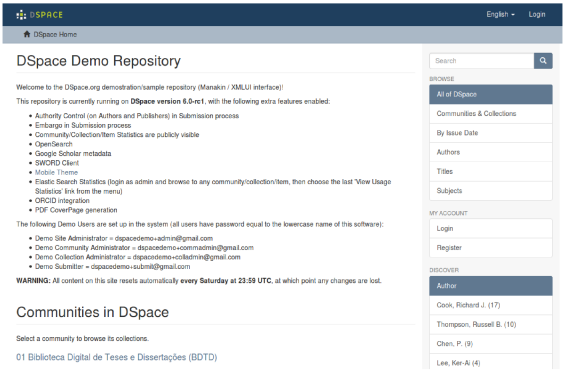

Figura 4. Página inicial em interface XMLUI do DSpace.

1.7. Detalhes Técnicos do DSpace

1.7.1. Metadados

DSpace faz o uso do padrão Dublin Core para descrição de metadados de seus itens, havendo um mínimo de três campos requeridos para a composição de um item: título; língua; e a data de submissão. Todos os outros campos, como resumo, palavras-chaves, detalhes técnicos, direitos autorais, entre outros, são opcionais. Estes metadados são visualizados dentro de um item no DSpace[Smith et al. 2003].

1.7.2. Interface

A interface disponível para usuários é *web-based*, contendo várias sub-interfaces. Uma para submetedores de itens e/ou qualquer processo que esteja envolvido com submissão;

uma para usuários finais que buscam pela informação; e outra para administradores do sistema[Smith et al. 2003].

A interface para usuários finais, ou interface pública, contém campos de busca com retorno de valores para navegação ou valores de metadados. Uma vez que o item é localizado no sistema, o usuário é capaz de fazer o download do arquivo em anexo (se disponível e se houver) através de seu navegador web[Smith et al. 2003].

Ainda a respeito da interface web, ela pode ser dividida em duas opções de implementação, a JSPUI (Java Server Pages User Interface) e XMLUI (eXtented Mark Language User Interface), com vantagens e desvantagens de cada uma delas, cabendo ao administrador ou equipe responsável pelo repositório decidir qual delas utilizar[Shintaku and Meirelles 2010].

A versão atualmente utilizada do XMLUI no DSpace é chamada de Manakin. Essa versão passa a fazer uso do framework Apache Cocoon(`http://cocoon.apache.org/`) para usar uma abordagem Simple API for XML (SAX). Para completar essa customização, Manakin usa temas(*Themes*) para estilo de conteúdo e pacotes chamados de *Aspects* para modularizar a geração do conteúdo[Phillips et al. 2005].

Não estando presente inicialmente, a interface XMLUI apareceu a partir da versão 1.5 provendo grandes melhorias[Shintaku and Meirelles 2010]. A principal delas é a possibilidade de customização, oferecendo ao usuário a escolha de adotar características distintas para comunidades, coleções e itens. Outras inovações são a separação da camada de negócio da interface, internacionalização e localização de conteúdos e compatibilidade com a então atual interface baseada em JSP[Wolf et al. 2013].

Como desvantagens, podemos citar a maior complexidade para alteração e manutenção. Como utiliza folha de estilos e programas conversores (XSL), exige profissionais mais especializados e de maior conhecimento em edição de *front-end* de sistemas web[Shintaku and Meirelles 2010].

A figura 4 mostra a interface Manakin XMLUI, à qual foi aplicado o tema Mirage, padrão da instalação da versão 6.0.

A arquitetura do Manakin consiste de 3 camadas. Em cada camada, níveis de componentização estão presentes, possibilitando um desenvolvimento em paralelo. Progressivamente, é requerido menor experiência para desenvolvimento, conforme desce-se nas camadas[Phillips et al. 2005].

- Camada de desenvolvimento Java/Cocoon: Requer conhecimento em Java e Cocoon framework e possibilita realizar qualquer customização funcional para o DSpace;
- Camada de tema XML/XSL: Exige conhecimento XML e XSLT, embora não precisando mais da linguagem Java. Nesta camada, informações processadas pelo DSpace podem ser manipuladas antes de mostrar ao usuário final;
- Camada de estilos HTML/CSS: Necessita apenas de conhecimento XHTML básico e permite modificar o estilo das informações que são mostradas diretamente para o usuário final.

1.7.2.1 Apache Cocoon

Cocoon descreve-se como um framework para desenvolvimento web construído sob os conceitos de *Separation of Concerns* (SoC) e arquiteturas baseadas em componentes. Juntando com a API SAX, estas características reúnem argumentos favoráveis necessários para uso desta tecnologia[Phillips et al. 2005].

O princípio do SoC habilita o desenvolvimento em paralelo das aplicações. Cada desenvolvedor pode concentrar-se em uma única parte do projeto sem alteração ou influência de outro. Como o desenvolvimento do DSpace é largamente provido por contribuições de comunidades de programadores, o uso do SoC se veio como uma facilidade, pois antes haviam muitos conflitos de projetos decorrentes das amarrações que interfaces JSP possuiam[Phillips et al. 2005].

1.7.2.2 *Sitemap*

O *sitemap* é um cojunto de documentos XML que descreve a configuração de todos os componentes Cocoon. O *sitemap* contém duas principais partes: a definição da seção dos componentes que descreve cada tipo de componente, e uma seção de pipeline que define como estes componentes são arranjados. Pode-se dizer que o *sitemap* é o coração do framework[Phillips et al. 2005].

1.7.2.3 Temas

Como já dito, os temas contribuem para a estilização do conteúdo gerado e disponível pelo Manakin. O formato é tipicamente em XHTML, entretanto, não existe impedimento para uso de outros formatos como PDF ou SVG, por exemplo. Temas são implementados como folhas de estilo XSL. Todos os conteúdos digitais usados para a estilização do tema são empacotados juntos para criar assim uma portabilidade. Temas podem ser configurados para ser aplicados à uma comunidade/coleção particular ou em muitas ao mesmo tempo. Também podem ser aplicados em uma única página arbitrária[Phillips et al. 2005].

1.7.2.4 *Aspect*

No paradigma de programação orientada a aspectos (Aspect-Oriented Programming (AOP)), programas são separados em partes distintas sobrepondo-se o mínimo possível. Estas partes são chamadas de *aspects*, e entrelaçam-se para formar o programa. No Manakin, estes *aspects* se combinam para formar todas as suas características. Essa é mais uma estrutura que potencializa a interface, pois todos os aspectos estão estruturados separadamente[Phillips et al. 2005].

A ordem com que os aspectos são executados é determinado pelo arquivo de configuração. *aspects.xml*. Nesse caso, aspectos presentes na configuração são empacotados juntos com o código-fonte, *sitemaps* e outros recursos que possam ser requeridos. Este empacotamento faz com que gere-se portabilidade, sendo fácil transportar de um DSpace para outro causando um mínimo de conflito[Phillips et al. 2005].

Em sua versão atual, os aspectos estão disponibilizados da seguinte forma[Duraspace]:

- ViewArtifacts: Responsável pela exibição de um item individual;
- BrowseArtifacts: Reponsável pela exibição de diferentes opções da navegação;
- SearchArtifacts: Responsável pela exibição de diferentes campos de busca (não usado por padrão);
- Administrative: Responsável pelo conjunto de ferramentas administrativas que o sistema contém;
- E-Person: Responsável pela adminsitração de usuários e todo seu contexto;
- Submission: Responsável pela submissão de novos itens no DSpace;
- Statistics: Responsável pela exibição das estatísticas do DSpace;
- Workflow: Responsável pelas tarefas de fluxo de trabalho que o DSpace apresenta (não usado por padrão);
- XMLWorkfflow: Adicionado a partir da versão 1.8 e substitui desde então o aspecto Workflow;
- Discovery: Também adicionado posteriormente, substitui os aspectos relacionados a buscas até então;
- SwordClient: Referente ao sistema SWORD presente no DSpace e que não será discutido neste trabalho;
- XMLTest: Aspecto criado para dar assitência ao desenvolvedores contendo opções de debugging;
- ArtifactBrowser: Aspecto depreciado. Foi divido em ViewArtifacts, BrowseArtifacts e SearchArtifacts a partir da versão 1.7.

1.7.3. Tecnologias usadas

DSPace foi desenvolvido para ser código aberto, de forma que instituições e organizações possam usufruí-lo com uma quantidade mínima de recursos. O sistema é desenvolvido para rodar em plataforma UNIX abrangendo várias outras ferramentas e middlewares escritos pelos desenvolvedores[Smith et al. 2003].

Todo o código original é escrito na linguagem de programação Java. Outras partes essenciais para o sistema incluem o sistema de base de dados relacional PostgreSQL, um servidor web (Apache), um servidor Java (Tomcat) e outras bibliotecas de sistema como Jena (RDF) e OAICat[Smith et al. 2003].

Vale ressaltar que enquanto DSpace é um software aberto e gratuito, seus desenvolvedores (MIT e HP) não oferecem suporte formal para seus usuários. É assumido que instituições que decidirem usar o software possuam recursos suficientes para a manutenção e configuração deste[Smith et al. 2003].

Os softwares necessários para a execução e instalação do DSpace serão apresentados e discutidos de forma mais aprofundada no capítulo 4.

1.7.4. Requisitos de Hardware

De acordo com o manual do DSpace [DSpace 2016b], a recomendação varia muito dependendo do tamanho que o repositório pretende atingir. Para uma aplicação mínima do

software, requer-se, 2-3GB de RAM e 20GB de armazenamento, enquanto que para uma aplicação pesada pede-se 8GB de RAM com 1TB de armazenamento.

1.7.5. Sistema Operacional

O DSpace pode ser implantando tanto no Sistema Operacional *UNIX-like* ou Microsoft Windows. Em sistemas operacionais *UNIX-like* já se contém algumas dependências para a instalação pré-instaladas ou que são facilmente instaladas via atualizações em seus repositórios, diferentemente do Microsoft Windows onde todos os programas precisam ser instalados manualmente[DSpace 2016b].

1.7.6. Arquitetura de sistema

A arquitetura do DSpace é dividida em 3 camadas: armazenamento, lógica de negócios e aplicação. A camada de armazenamento é implementada usando um sistema de arquivos e gerenciada pelas tabelas do banco de dados do PostgreSQL. A de lógica de negócios é onde as funcionalidades específicas do DSpace residem, incluindo o *workflow*, a gerência de conteúdo, a administração, módulos de busca e navegação. Cada módulo dispõe de uma API que permite que os usuários possam ter as funcionalidades que desejam. Por fim, a camada de aplicação cobre toda a interface do sistema[Smith et al. 2003].

A figura 5 ilustra a arquitetura do sistema do DSpace.

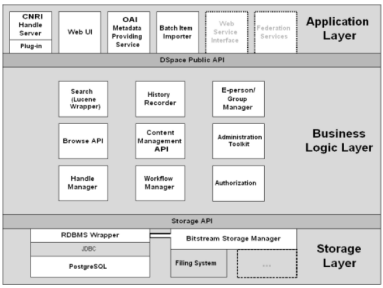

Figura 5. Arquitetura de sistema do DSpace. Fonte [Smith et al. 2003]

1.7.7. *Handles*

Um dos objetivos da preservação digital é possibilitar que se busquem e achem conteúdos depositados nos repositórios digitais a longo prazo. Para isso, é crucial que itens citados encontrados em artigos, por exemplo, possam ser encontrados efetivamente por longos períodos de tempo. Para esse fim, DSpace escolheu implementar CNRI Handles (`https://www.handle.net/`) como identificadores de persistência de cada item o qual embui-se das tarefas de gerências de itens de repositórios cadastrados ao redor de mundo[Smith et al. 2003].

Mais especificamente, no DSpace, Handles são associados a comunidades, coleções e itens. O Conteúdo digital anexado ao item não possui Handle, uma vez que esses podem ser mudados devido a alguma atualização, por exemplo[DSpace 2016b].

1.8. Sistemas de Recomendação

Sistemas de recomendação dentro do contexto de RDs têm como função recomendar obras, baseados em um conjunto de informações específicos, que neste caso, advém de metadados de um item pertencente ao RD, e que sejam potencialmente interessantes ao usuário. De modo geral, um sistema de recomendação se utiliza de três operações: o perfil do usuário, que manterá informações necessárias à identificação das preferências do usuário; a filtragem de informação, que seleciona os itens relevantes ao usuário utilizando para isso os perfis de usuário e dados de todos os itens; e o ranqueamento dos itens, servindo para listar de forma ordenada por relevância as obras escolhidas pelo sistema de recomendação[Salles and Willrich 2015].

1.9. Tipos de sistemas de recomendação

1.9.1. Abordagem clássica

Conforme descrito no livro *Recommender Systems Handbook* [Ricci et al. 2011], há 6 tipos de abordagens clássicas de sistemas de recomendações:

- Baseado em Conteúdo: O sistema aprende a recomendar com base em itens similares na preferência do mesmo usuário no passado;
- Filtragem Colaborativa: Como o nome sugere, a recomendação é calculada com base no que outros usuários preferiram no passado;
- Demográfica: Retorna itens baseados no perfil demográfico do usuário. Por exemplo, usuários recebem recomendações de itens baseados em sua língua ou país;
- Baseado em Conhecimento: Recomenda itens baseados em um domínio específico de conhecimento sobre como certas características de itens satisfazem as necessidades e preferências do usuário;
- Filtragem Híbrida: Combina duas ou mais técnicas de sistema de recomendação.

1.9.2. Baseado em grafos

O objetivo da representação em grafos é a otimização do processo de recomendação. No caso da Filtragem Colaborativa, por exemplo, pode-se modelar a recomendação na forma de um grafo bipartido onde os nodos representariam usuários e itens com arestas interligando-os. Em uma técnica baseada em conteúdo, os nodos de um grafo bipartido podem representar itens e valores com ligações entre eles na forma de arestas. Por último, em uma filtragem híbrida, é possível criar um modelo com nodos representando usuários, itens e metadados em um grafo, dessa vez, tripartido. A figura 6 mostra uma ideia dessa último modelo[Salles and Willrich 2015].

1.9.3. Baseado em ontologias

Uma ontologia é uma representação formalizada e não ambígua do conhecimento de um domínio, definindo conceitos, propriedades e relacionamentos[Salles and Willrich 2015].

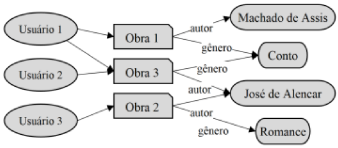

Figura 6. Recomendação em filtragem híbrida em modelo de grafo. Fonte [Salles and Willrich 2015]

Dessa maneira, seu uso em sistemas de recomendação consiste em explorar o relacionamento entre conceitos e a semântica das propriedades, podendo obter interesses de usuários que não são diretamente observados. Para exemplificar, pode-se citar um usuário que acessa várias obras literárias de autores de um determinado estado. Esse comportamento pode indicar que este usuário pode ter interesses por obras advindas desse estado. Em sistemas de recomendações tradicionais, esse comportamento não é percebido, pois concentra-se apenas em propriedades de itens já acessados[Salles and Willrich 2015].

1.10. Trabalhos Relacionados

Existem diferentes soluções de recomendações para repositórios digitais. Três delas merecem destaque: frameworks de recomendação, add-ons para RDs; e Web service [Salles and Willrich 2015].

Na primeira, no framework, os mais populares são *Apache Mahout*, *LensKit* e *MyMediaLite*. O uso de framework de recomendação integrados a RDs permite o reuso de componentes de software escaláveis e eficientes para recomendação. No entando, usar frameworks resulta em uma implementação de recomendação bastante acoplada com à estrutura de dados dos RDs, além de ser uma solução específica para um RD, não possibilitando sua generalização e transporta para RDs federados.

A segunda solução , por *add-ons*, já é uma forma de reduzir o esforço de implantação de serviços de recomendação em RDs abertos. No caso específico do DSpace, um trabalho relacionado nesse sentido é o *add-on* proposto em [Elliott et al. 2008]. Neste trabalho, os autores apresentam o Quambo, que adiciona contexto de pesquisas, itens favoritos e um próprio sistema de recomendação. Porém como se trata de um *add-on*, esta alternativa limita o reuso em RDs de uma determinada solução e assim como a solução em framework, não oferece suporte a RDs federados. Sua implementação é feita toda dentro da estrutura do DSpace deixando claro seu alto acoplamento com o sistema.

Por último, a solução por Web service apresenta a estrutura de maior desacoplamento com os RDs em si, e portanto se mostra a melhor opção em caso de RDs federados pois os dados e usuários podem ser combinados pelo servidor. Para esta solução, podemos citar os trabalhos de [Anjorin et al. 2012], [Beham et al. 2010] e [Lemos et al. 2012]. Em [Lemos et al. 2012] os autores propõem um sistema de recomendação de fotos baseado em *RestFul Web Service*, permitindo que dispositivos móveis de baixa capacidade de processamento possam usar o serviço. No entando o trabalho visa apenas um único domínio de aplicações (fotos), e os dados utilizados para a recomendação são obtidos de bases

de fotos específicas. Em anjorin2012framework, os autores propõem um sistema de recomendação de objetos de aprendizagem possibilitando o seu uso em plataformas de ensino. Mas, novamente, o domínio se limita a apenas objetos de aprendizagem. Em [Beham et al. 2010], é introduzido o sistema APOSTDL, que publica recomendações de especialistas. Para isso, o sistema se baseia em uma ontologia de domínio que representa tarefas em um domínio de aprendizagem. O perfil do usuário é implicitamente capturado observando as tarefas realizadas pelo usuário [Salles and Willrich 2015].

1.11. Sistema de Recomendação Adotado

Em seu trabalho,[Salles and Willrich 2015] estudam várias propostas já feitas como embasamento para a sua, e diferentemente dos trabalhos descritos na seção anterior, esta, independente de domínio, possibilitando especificar os conhecimentos do domínio relacionado aos itens a recomendar via a especificação de uma base de conhecimento ontológica.

Portanto, neste serviço planejado, o sistema de recomendação proposto por [Salles and Willrich 2015], são identificados quatro principais atores participantes (ilustrados na Figura 7):

- Administrador do contexto: Responsável por cadastrar o contexto (dados usados pelo serviço) no qual o serviço de recomendação será oferecido e de especificar a base de conhecimento (Knowledge Base (KB)) inicial;
- Usuário: São os usuário que acessam o repositório digital e recebem recomendações
- Repositório Digital: Atua como cliente do serviço de recomendação. Tem por função notificar acesso realizado pelos usuários e solicitar a recomendação ao sistema via Web service;
- Serviço Web de recomendação: Serviço de recomendação propriamente dito. Formado pelo Gestor de contexto e Recomendador que implementa os algoritmos de recomendação. Dentro do servidor, as informações são mantidas em Banco de Dados chamados de BD de contextos que contém os dados gerais dos contextos já cadastrados. BDs dos contextos mantém as bases de conhecimento dos contextos já cadastrados.

1.11.1. Modelagem conceitual dos Dados

Para especificar a base de conhecimento, foi adotada a linguagem OWL (*Web Ontology Language*) [Group 2012]. Com isso, descreve-se então uma ontologia de aplicação, uma ontologia de domínio do contexto e um ontologia dos indivíduos. A ontologia de domínio do contexto especifica conceitos e relacionamentos associados ao domínio de conhecimento relativos aos conteúdos digitais de um contexto de recomendação. Como exemplo de ontologia de domínio de contexto, podemos citar o domínio de obras literárias, onde obras, autores e gêneros literários representam essa ontologia[Salles and Willrich 2015].

A ontologia de aplicação é um ontologia desenvolvida para uma aplicação específica. Portanto, para este trabalho, é criada uma ontologia de aplicação chamada de **RecOnt**, ou ontologia de recomendação. A figura 8 apresenta a ontologia de recomendação. A *Audience* representa o grupo de usuários, que é o foco da recomendação. *Item*

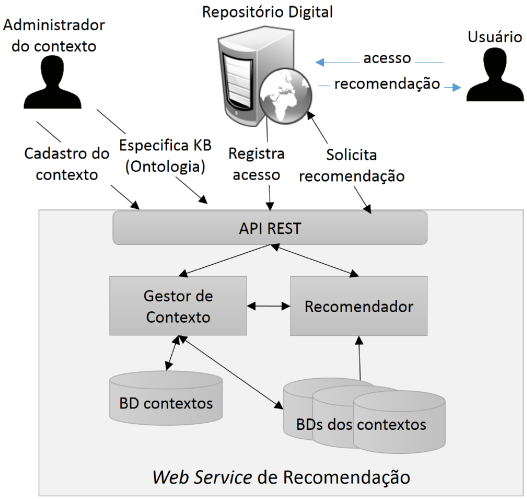

Figura 7. Interação dos atores no sistema.

significa o item a ser recomendado. E o *Factor of Interest*, simboliza determinado conceito importante a Item. No caso do DSpace com o padrão *Dublin Core*, um conceito relevante poderia ser um grupo de autores e palavras-chaves de um mesmo item[Salles and Willrich 2015].

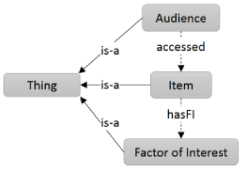

Figura 8. Ontologia de recomendação RecOnt[Salles and Willrich 2015].

É importante observar que esses conceitos descritos são representados por URIs, fazendo com que o sistema de recomendação seja desacoplado do repositório digital, no caso deste trabalho, do DSpace. Assim, o sistema provê privacidade ao usuário, pois não captura nenhuma informação pessoal sobre o tipo de conteúdo que ele está acessando. Finalmente, podemos dizer que a presente proposta se torna uma solução de propósito geral, já que se adapta facilmente a qualquer domínio do conhecimento[Salles and Willrich 2015].

1.11.2. Interface de Acesso

A interface de acesso do serviço é via Web service do tipo REST, disponibilizando dados em formato JSON. Escolheu-se este tipo por se tratar de um formato simples e de fácil interpretação e manipulação por humanos e máquinas independente de linguagem de programação e/ou framework e sistema operacional, já que os dados precisarão passar por algum tipo de processamento, não só Web como localmente no servidor[Salles and Willrich 2015].

1.11.3. Banco de dados orientado a grafos

Os dados coletados pelo sistema de recomendação são armazenados e organizados em forma de grafo em um banco de dados orientado a grafos. Como sistemas de recomendação estão relacionados à escalabilidade e considerando resultados promissores de testes já realizados, optou-se pelo uso do banco de dados Neo4j para tal finalidade[Salles and Willrich 2015].

1.12. Considerações finais

Embora o foco do projeto seja o software DSpace, havendo então outras maneiras de construção de sistemas de recomendações que não a proposta neste trabalho, por exemplo, a utilização de *add-ons* baseados em trabalhos anteriores [Elliott et al. 2008], a estratégia escolhida é a de um sistema de recomendações em Web service baseado em ontologias. Desta maneira, é possível oferecer um baixo acoplamento do sistema como um todo, pois toda a lógica de negócio da recomendação será computada em um servidor diferente, cabendo ao RD apenas atuar como cliente, simplesmente conectando-se ao serviço já mencionado. Além disso, a solução proposta oferece formas de prover a recomendação em federações, pois os dados e usuários podem ser combinados pelo servidor[Salles and Willrich 2015].

1.13. Sistema de Recomendação para DSpace

Este capítulo descreve a implantação do sistema de recomendação baseado em Web service no DSpace . A seção 4.1 descreve os componentes existentes do sistema bem como suas especificações usadas para a implantação, enquanto que a seção 4.2 atenta em descrever como são definidas as ontologias que comporão o sistema de recomendação. A seção 4.3 visa especificar a forma que o sistema atuará no DSpace, subseguindo com a seção 4.4 e 4.5 que se ocupará da parte de implementação e avaliação da integração respectivamente.

1.14. Componentes do Sistema de Recomendação

Os principais componentes do Sistema de Recomendação proposto são (Figura 9): cliente, servidor do repositório e serviço de recomendação.

1.14.1. Cliente

No sistema, o cliente é responsável pelo registro e acesso de recursos disponíveis no repositório. Para tal, sua atuação utiliza-se somente de um navegador para acesso à plataforma DSpace.

Figura 9. Principais componentes do sistema de recomendação proposto

1.14.2. Repositório DSpace

Neste caso, o Repositório é constituído da plataforma para repositórios digitais DSpace. A versão do software utilizada neste projeto foi a 6.0 e sua instalação obedeceu os passos e instruções descritos nesta seção.

A máquina usada para o desenvolvimento foi um Intel Core i5 com 16GB de RAM e disponibilidade de 500GB de armazenamento. O sistema operacional (SO) adotado nesta máquina foi a distribuição Linux Ubuntu na versão 14.04.

Como etapa preliminar a instalação do DSpace propriamente dito, foi necessária a instalação dos seguintes pacotes de softwares [DSpace 2016b]:

- *Java SE Development Kit*: A documentação do DSpace indica as opções de máquina virtual Oracle Java JDK 1.7+ ou OpenJDK 1.7+. Neste projeto foi adotada a implementação livre a gratuita do Java OpenJDK $1.8.0_111$;
- Apache Maven: Maven é necessário no primeiro estágio de compilação do DSpace. A utilização do Maven permite criar módulos no código-fonte, fazendo com que se possa modificar e compilar partes do código-fonte sem a necessidade de construir todo ele novamente para cada alteração feita. Neste projeto foi utilizado a versão 3.0.5 do Maven. Até a presente versão do DSpace (6.0), os módulos criados pelo Maven são: dspace-api; dspace-jspui; dspace-oai; dspace-rdf; dspace-rest; dspace-services; dspace-solr; dspace-sword; dspace-swordv2; dspace-xmlui; e dspace-xmlui-mirage2;
- Apache Ant: Apache Ant é requerido como segundo estágio do processo de compilação. Ant é usado uma vez que os pacotes são construídos pelo Maven no diretório *[dspace-source]/dspace/ target/dspace-installer*. Neste projeto foi instalado o Apache Ant 1.9.3;
- Sistema Gerenciador de Banco de Dados: A Documentação do DSpace indica a possibilidade de utilizar o PostgreSQL 9+ ou o Oracle 10+ (nos dois casos sendo necessário a habilitação do suporte à Unicode). Neste projeto foi adotado o SGBD PostgreSQL. A partir da versão 6.0 do Dspace requer uma versão maior ou igual que a 9.4 com a extensão pgcrypto instalada. Portanto, para atingir estes requisitos, usou-se a versão 9.4.10;
- Apache Tomcat: A versão adotada neste projeto foi a Tomcat 7.0.52, sendo que a documentação do DSpace recomenda a utilização da versão a partir da 7.0.30, visto que versões abaixo desta apresentam problemas de estouro de memória, exi-

gindo grandes quantidade de uso de memória somente para o Tomcat.

Após a configuração dos softwares requeridos no sistema operacional, foram realizados os seguintes passos para instalação do DSpace, conforme o manual [DSpace 2016b]:

- Criação de um usuário DSpace: **useradd -m [usuário]** (Este usuário precisa ser o mesmo que o Tomcat irá usar);
- *Download* da versão 6.0 do DSpace disponível em seu website;
- Criar usuário e banco de dados em um sistema de gerenciamento de banco de dados escolhido;
- Configuração das variáveis iniciais do sistema disponíveis no arquivo *build.properties*;
- Criação do diretório com permissão de usuário dspace (ou outro nome escolhido) na criação do usuário com base no caminho escolhido no passo anterior;
- Uso do comando **mvn package** dentro do diretório raiz do código-fonte do DSpace para compilação dos módulos. (Este comando utiliza o arquivo *pom.xml* para compilação);
- Após o primeiro passo da compilação, executou-se o comando **ant fresh_install** dentro do diretório *dspace-installer* criado em *[dspace-source]/dspace/target/dspace-installer* para a instalação dos módulos compilados pelo maven no sistema operacional;
- Realização do *deploy* da interface escolhida, disponível no diretório *webapps* do DSpace instalado, no Tomcat;
- Execução do Tomcat e acesso à URL configurada para acessar o sistema.

Conforme discutido anteriormente, existem duas opções para a implementação das interfaces Web do DSpace: JSPUI (Java Server Pages User Interface) e XMLUI (eXtented Mark Language User Interface). Neste projeto foi adotada o XMLUI, pois por razões já enunciadas no capítulo 2, ela apresenta vantagens quando leva-se em questão sua customização.

É relevante destacar que este projeto visa incorporar a funcionalidade de Sistema de Recomendação no DSpace, e não uma simples personalização das interfaces Web (por exemplo, visando atender as identidades visuais de uma determinada organização). A incorporação de novas funcionalidades no DSpace foi um desafio a parte, devido à carência de documentação disponíveis aos desenvolvedores.

Para incorporar as funcionalidades de um Sistema de Recomendação no DSpace, é necessário o uso de sua API e obedecer ao padrão de projeto adotado pelo DSpace. Assim, uma página de visualização necessitará de um novo *aspect* da interface possuindo métodos com assinaturas específicas. Aos que se pode observar, temos os métodos *addBody*, *addOptions*, *addPageMeta*, e *addUserMeta*. Cada método não é excludente ao outro, podendo todos estarem presentes ou não. Esses métodos são responsáveis por receber objetos de um determinado segmento das informações que serão processadas pelas camadas posteriores.

Para a construção da lógica de negócio da solução proposta, ou seja, o tratamento dos dados vindos do DSpace e enviados para o sistema de recomendação, a classe no DSpace pode ser construída sem a adição desses métodos mencionados no parágrafo anterior, estando amarrada somente à sua API.

1.14.3. Serviço Web de Recomendação

É a aplicação que executa a lógica de negócio da recomendação. Seus algoritmos não serão discutidos neste trabalho devido à sua complexidade.

O serviço foi desenvolvido na linguagem Java e utilizado a OWL-API para a importação das ontologias.

Seu hardware é formado por um servidor HP ProLiant DL180G5, processador Intel Quad Core Xeon E5404, 8GB de memória RAM e 500GB de armazenamento. O SO em questão é a distribuição Linux Ubuntu na versão 14.04.

Os softwares instalados e necessários para seu funcionamento são:

- Apache Tomcat 7.0.52;
- Java Oracle 1.8.0_91b14;
- Neo4J 2.3.7.

1.15. Ontologia de Domínio e de Contexto

Para a instalação da RecOnt no domíno de repositórios DSpace, deve-se definir uma ontologia de domínio e uma ontologia de contexto. Esta seção ilustra como são definidas estas ontologias para implantação do sistema de recomendação em uma instalação DSpace.

A ontologia de domínio define conceitos gerais sobre o domínio dos recursos disponibilizados no repositório DSpace. Para este trabalho, adotou-se uma ontologia de domínio genérica, considerando conceitos genéricos existentes em repositório DSpace, chamada *DCResource*. A Figura 10 apresenta a *DCResource*. Os conceitos desta ontologia são:

- *Resource*: Representa um recurso no DSpace, sendo um recurso digital como um artigo, tese, dissertação, etc. Um *Resource* possui um ou mais autores (*hasAuthor*) e uma ou mais palavras-chave (*hasSubject*);
- *Author*: Representa um autor do recurso;
- *Subject*: Representa uma palavra-chave associada ao recurso.

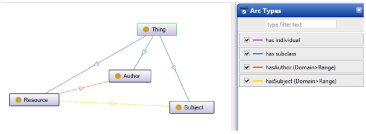

Figura 10. Ontologia DCResource.

As classes *Subject* e *Author* foram escolhidas pois os autores deste trabalho consideram estes os metadados DC mais relevante para capturar as preferências/interesses do usuário de um repositório DSpace que armazene qualquer tipo de recursos.

A ontologia de contexto adotada é DCContext, apresentada na Figura 11. Ela define que o item a recomendar é da classe *Resource*, e que os fatores de interesse são *Author* e *Subject*

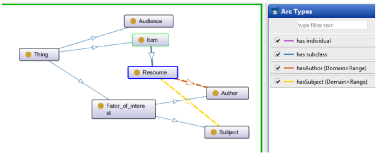

Figura 11. Ontologia DCContext.

1.16. Especificação do Sistema de Recomendação integrado ao DSpace

Esta seção visa especificar, utilizando diagramas UML, os locais e formas de recomendação a serem integradas no repositório DSpace.

1.16.1. Registro de Acessos a Recursos

O sistema RecOnt disponibiliza alguns algoritmos de recomendação. A recomendação por popularidade não requer a identificação do usuário acessando o recurso. Já na recomendação por filtragem colaborativa e baseada em conteúdo implementada atualmente exigem a captura do histórico do usuários. Este histórico contém os recursos acessados pelo usuário, bem como os seus metadados.

O Diagrama de Sequência da Figura 12 apresenta o acesso de um recurso por parte de um usuário logado. O método de registro de acesso implementado deve realizar as seguintes ações:

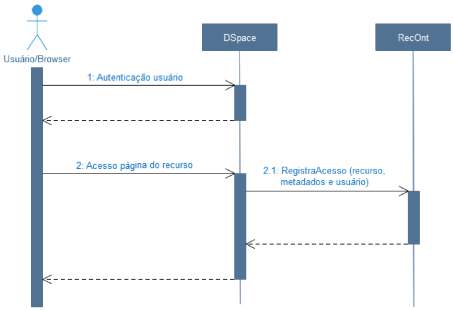

Figura 12. Diagrama de sequência de um registro de um recurso baseado em conteúdo

- Coletar o identificador do usuário no DSpace;
- Coletar os metadados do recursos acessados considerados relevantes para capturar as preferências do usuário. Neste projeto,considerou-se os metadados autor (dc.contributor.author) e assuntos (dc.subject);

- Representar o item e seus metadados em OWL com base na ontologia de contexto DCContext;
- Chamar o serviço web de registro de acesso da RecOnt.

No caso de um acesso a um recurso por um usuário não logado, o procedimento de registro de acesso é o mesmo, com exceção que é utilizado um usuário especial, chamado *Anonymous*, apenas para fins de recomendação por popularidade.

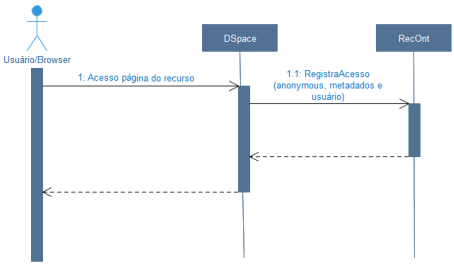

Figura 13. Diagrama de sequência de um registro de um recurso baseado em popularidade

A requisição do Web service RecOnt para registro de acesso a recursos deve seguir o seguinte padrão:

- **Método POST**:
  Exemplo: `http://localhost:8080/{ApplicationName}/{WebService}/access/apiKey/Context/actionTime`
  Onde temos os seguintes parâmetros para a construção da estrutura da URL como no exemplo acima:
  - ApplicationName: Nome da aplicação;
  - WebService: Classe contendo os métodos do Web service;
  - access: Método que será executado.
  - apiKey: Token responsável pela identificação do repositório cliente;
  - Context: Identificação do contexto do repositório, podendo ser 1 ou mais;
  - actionTime: Tempo em que o acesso ocorreu.

1.16.2. Apresentação da Recomendação

Existem dois cenários de apresentação da recomendação para os usuários do repositório. O primeiro é para o caso de usuários não autenticados no sistema, e o segundo, para usuários autenticados.

O Diagrama de Sequência da Figura 14 apresenta o acesso à página principal do DSpace por parte de um usuário não logado, ou seja, um usuário anônimo. O método de recomendação implementado deve realizar as seguintes ações:

- Solicitar a recomendação por Popularidade ao RecOnt, utilizando o WebService RecOnt. Conforme previsto na API, esta chamada deverá conter o identificador do usuário *Anonymous* e o identificador do tipo de recomendação solicitada;
- Receber a lista de recursos recomendados na forma de uma estrutura JSON. Cada recurso será identificado pelo seu handle;
- Geração dinâmica da página principal contendo a lista de recomendação na forma de uma div HTML. Para tal, com base no handle, o sistema deverá recuperar o título e os autores do recurso.

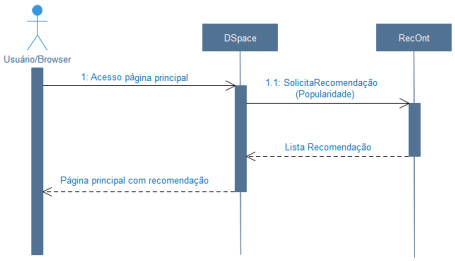

Figura 14. Diagrama de sequência de um registro de um recurso baseado em popularidade

A justificativa da recomendação por popularidade é que, em um primeiro acesso ao repositório, o sistema não tem ainda nenhum indício do que o usuário procura, então a filtragem colaborativa seria mais adequada, pois veria o que os outros usuários já acessaram. Quando o usuário entra na página de um recurso, a recomendação por conteúdo é mais adequada, pois iria recomendar itens parecidos com o que o usuário acessa naquele momento.

O Diagrama de Sequência da Figura 15 apresenta o acesso à página principal do DSpace por parte de um usuário autenticado. O método de recomendação implementado deve realizar as seguintes ações:

- Solicitar a recomendação por Filtragem Colaborativa ou Baseada em Conteúdo (de escolha do desenvolvedor) ao RecOnt, utilizando o WebService RecOnt. Conforme previsto na API, esta chamada deverá conter o identificador do usuário e o identificador do tipo de recomendação solicitada;
- Receber a lista de recursos recomendados na forma de uma estrutura JSON. Cada recurso será identificado pelo seu handle;
- Geração dinâmica da página principal contendo a lista de recomendação na forma de uma div HTML. Para tal, com base no handle, o sistema deverá recuperar o título e os autores do recurso.

O método da RecOnt para solicitação de recomendação deve seguir o seguinte formato:

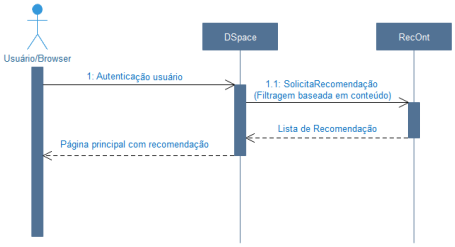

Figura 15. Diagrama de sequência de um acesso de um recurso baseado em conteúdo

- **Método GET**:
  Exemplo: `http://localhost:8080/{ApplicationName}/{WebService}/rec/apiKey/{user}/nresult/rectype`
  Onde temos os seguintes parâmetros para a construção da estrutura da URL como no exemplo acima:
  - ApplicationName: [alfanumérico] Nome da aplicação;
  - WebService: [alfanumérico] Classe contendo os métodos do Web service;
  - rec: Método que será executado.
  - apiKey: [alfanumérico] Token responsável pela identificação do repositório cliente;
  - user: [alfanumérico] Usuário alvo da recomendação;
  - nresult: [inteiro] Numero de resultados;
  - recType: [inteiro] Tipo da recomendação (1: Por popularidade, 2: Baseada em conteúdo, 3: Colaborativa).

1.17. Implementação

Antes de mais nada, é necessário deixar claro que foi imprescindível o uso de um ambiente de desenvolvimento integrado (IDE) para o desenvolvimento da proposta. Para isto, escolheu-se o IntelliJ IDEA da empresa JetBrains (`https://www.jetbrains.com/idea/`). Sua versão completa disponibiliza uma interface bem adaptada para *debugs* on-the-fly em aplicações como o Tomcat.

O sistema implementado utiliza-se de duas classes Java inseridas no aspecto *viewArtifacts* localizado dentro do módulo dspace-xmlui:

- *AccessRegister*: Responsável pela coleta dos metadados do recurso, construção da ontologia e envio das informações via HTTP;
- *RecOnt*: Encarrega-se da solicitação dos dados no formato JSON a partir do servidor de recomendação e trata-os para a construção de recursos dentro do DSpace.

1.17.1. AccessRegister

A classe AccessRegister contém um método principal chamado *callWebService* que se encarrega de abrir uma conexão HTTP e construir a ontologia passando os metadados como variáveis, conforme mostrada na figura 16. Essa construção é atribuída à uma *string* onde faz-se sucessivas concatenações entre o texto imutável e as variáveis coletadas quando o usuário acessa o item. Essas variáveis são coletadas através de métodos *getters* a partir de uma instanciação da classe AccessRegister dentro da classe nativa do DSpace *ItemViewer*, responsável por controlar a exibição de um item no DSpace. A assinatura do método recebe como parâmetros as *string* dos metadados coletados.

```
{
  public void callWebService(String author, String subject,
     String handle, String eperson)
}
```

Figura 16. Ontologia resultante com os metadados concatenados.

O método *getAuthor*, que pode ser observado abaixo, utiliza-se da classe *DSpaceObject* assim como o *getSubject* que contém a mesma lógica em seu código. No método *getEperson*, criado para retornar o usuário atual, utiliza-se a classe *Context* onde se está localizado o usuário enquanto que *getHandle* faz-se o uso das duas classes para o retorno do *Handle* do item no DSpace.

Abaixo, são mostradas essas três variações:

```
{
  public String getAuthor(DSpaceObject dso) throws
     SQLException {

      List<MetadataValue> author =
         itemService.getMetadataByMetadataString((Item)
         dso,"dc.contributor.author");
      this.author = author.get(0).getValue();
```

```
        this.author = this.author.replaceAll("\\s","");
           //Remove espacos em brancos o qual a ontologia no
           aceita.
        return this.author;
    }
}
```

Neste método podemos observar que é criada uma lista para armazenar a *string* autor através do método *getMetadataByMetadataString* da classe *ItemService*. Em seguinte executa-se uma ação de remoção de espaços em branco, pois a ontologia não funcionará se existir espaços em branco nas variáveis.

```
{
  public String getEperson(Context context) throws
     SQLException {
      EPerson eperson = context.getCurrentUser();
      this.eperson = eperson.getID().toString();
      return this.eperson;
    }
}
```

Para coletar o usuário, é necessário o uso da classe *Context*, pois é só nela que é passado o usuário dentro do sistema. A classe *DSpaceObject* é construída para os metadados de um item além de outras informações não relevantes para este trabalho.

```
{
  public String getHandle(DSpaceObject dso, Context
     context) throws SQLException {
      ArrayList<String> identifiers =
         itemService.getIdentifiers(context, (Item) dso);
      this.handle = identifiers.get(0);
      this.handle =
         this.handle.substring(this.handle.lastIndexOf("/")+1);
      return this.handle;
    }
}
```

O *getHandle* em seu corpo, opera com o método *getIdentifiers* da classe *ItemService*. Para isso, usa-se tanto o *Context* quanto o *DSpaceObject* como seus parâmetros.

Na parte do envio do acesso, é criado a requisição HTTP pelo método POST com a construção do cabeçalho pelo código

```
{
      connection.setRequestMethod("POST");
      connection.setRequestProperty("User-Agent",
         "Mozilla/5.0 (X11; Linux x86_64)
         AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu
         Chromium/53.0.2785.143 Chrome/53.0.2785.143
```

```
            Safari/537.36");
        connection.setRequestProperty("Accept-Language",
            "pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4");
        connection.setRequestProperty("Content-Type",
            "application/xml");
    }
}
```

e enviado através da execução do trecho

```
{
        // Envio da requisicao POST
        connection.setDoOutput(true);
        DataOutputStream wr = new
            DataOutputStream(connection.getOutputStream());
        wr.writeBytes(ontology);
        wr.flush();
        wr.close();

        BufferedReader in = new BufferedReader(
             new
                InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
          response.append(inputLine);
        }
        in.close();
}
```

1.17.2. RecOnt

A classe *RecOnt*, por ser uma classe de exibição, faz o uso dos métodos *addBody* e *addPageMeta* já mencionados e estende a classe *AccessRegister*. Dessa forma, é montado, via API, uma divisão na página inicial responsável por listar itens que servirão para a recomendação. Dessa maneira, recupera-se os dados no formato JSON e fazendo o uso da biblioteca Gson, trata-se a string de retorno. Dessa string, é extraído o Handle que servirá como parâmetro para instanciar um objeto do DSpace pela classe *DSpaceObject* onde este representará um item na lista. Para repetir o processo, faz-se a utilização de um laço extraindo quantos Handles estiverem retornados na consulta ao servidor, criando assim uma lista de itens recomendados na página inicial.

O código tem duas seções que se comportam de forma bastante parecidas. Em um primeiro momento, verifica-se se o usuários está autenticado com o código

```
{
        if (context.getCurrentUser() != null) {
```

```
            String epersonId = null;
            epersonId =
               context.getCurrentUser().getID().toString();
}
```

Se o usuário estiver autenticado, é executado o trecho para a apresentação de itens por conteúdo. Caso contrário, por popularidade.

Um ponto importante do código é a lógica para a construção da lista dos itens de recomendação como mostrado abaixo.

```
{
            Division home = body.addDivision("site-home",
               "primary repository");
            Division recomendation =
               home.addDivision("site-recomendation",
               "secondary recomendation");
            recomendation.setHead("Itens sugeridos");

            JsonObject obj = root.getAsJsonObject()
                                 .getAsJsonObject("recommendation");
            obj = obj.getAsJsonObject("entry");
            String handle = "";
            for (int i = 0; i < obj.entrySet().size() - 1;
               i++) {
               handle = obj.get("value").getAsString();

               DSpaceObject dso =
                  handleService.resolveToObject(context,
                  "123456789/" + handle);

               ReferenceSet recomendationSet =
                  recomendation.addReferenceSet(
                     "site-recomendation",
                        ReferenceSet.TYPE_SUMMARY_LIST,
                     null, "recomendation");
               recomendationSet.addReference(dso);
}
```

Como observa-se, a partir do JSON enviado pelo sistema de recomendação, cria-se um laço onde para cada item instanciado é adicionado em um conjunto de recomendações. A partir disso, o DSpace se encarrega de criar a seção de ”Itens Sugeridos”na página inicial.

Nos exemplos a seguir, são mostrados os retornos do sistema de recomendação na chamada da classe *RecOnt*:

Exemplo JSON de retorno do Handle a partir de recomendação baseada em conteúdo:

```
{
  "audience":"38efcc71-46cd-4b62-9dbc-6f267f99d427",
  "recommendationType":"2",
  "recommendation":{
    "entry":{
      "key":"1",
      "value":"4"
    }
  }
}
```


Neste exemplo, 38efcc71-46cd-4b62-9dbc-6f267f99d427 representa a ID do usuário no DSpace. O tipo de recomendação é 2, portanto é um recomendação baseada em conteúdo, e seu Handle igual a 4.

Como exemplo de popularidade temos o seguinte retorno:


```
{
  "audience":"anonymous",
  "recommendationType":"1",
  "recommendation":{
    "entry":[
      {
        "key":"1",
        "value":"Resource"
      },
      {
        "key":"2",
        "value":"9"
      },
      {
        "key":"3",
        "value":"6"
      },
      {
        "key":"4",
        "value":"4"
      }
    ]
  }
}
```


Aqui, os Handles retornados são o 9, 6 e 4, formando uma lista de itens de recomendação por popularidade.

Após passar os Handles como parâmetros para instanciar os itens do DSpace, temos o resultado apresentado na figura 17.

Figura 17. Lista de recomendações por popularidade.

1.17.3. Sitemap

Por último, para o DSpace de fato criar a interface na posição da página desejada, é adicionado no arquivo *sitemap.xmap* as classes para fazer o mapeamento pelo Apache Cocoon, como mostrado nas marcações abaixo.

Declaração das classes no sitemap:

```
<map:transformer name="AccessRegister"
   src="org.dspace.app.xmlui.aspect.viewArtifacts.AccessRegister"/>
<map:transformer name="RecOnt"
   src="org.dspace.app.xmlui.aspect.viewArtifacts.RecOnt"/>
```

Inclusão da classe RecOnt na página inicial:

```
<!-- Display the DSpace homepage. This includes the
   news.xml file along with a list of top level communities
   in DSpace. -->
<map:match pattern="">
   <map:transform type="Include"
      src="resource://aspects/ViewArtifacts/dspace-home.xml"
      />
   <!-- DSpacePropertyFileReader will read the DSpace
      property file and place the selected properties value
      in this scope
   -->
      <map:act type="DSpacePropertyFileReader">
            <map:parameter name="dspace.dir"
               value="dspace.dir" />
            <map:transform type="Include"
               src="file://{dspace.dir}/config/news-xmlui.xml"
```

```
                />
        </map:act>

      <map:transform type="RecOnt"/> <!-- Classe RecOnt na
         pagina inicial-->
        <map:serialize type="xml"/>
</map:match>
```

Inclusão da classe AccessRegister na página do item:

```
<map:select type="browser">
<map:when test="spider">
  <map:select type="IfModifiedSinceSelector">
    <map:when test="true">
      <map:act type="NotModifiedAction" />
      <map:serialize />
    </map:when>
    <map:otherwise>
      <map:transform type="ItemViewer" />
      <map:transform type="AccessRegister"/> <!-- Classe
         AccessRegister na pagina pertencente ao item -->
      <map:serialize type="xml" />
    </map:otherwise>
  </map:select>
</map:when>
<map:otherwise>
  <map:transform type="ItemViewer" />
  <map:transform type="AccessRegister"/> <!-- Classe
     AccessRegister na pagina pertencente ao item -->
  <map:serialize type="xml" />
</map:otherwise>
</map:select>
```

1.18. Avaliação da Integração do Sistema de Recomendação e Testes

O objetivo deste trabalho não é o de avaliação de uma técnica específica de recomendação, a ser implantada no servidor RecOnt. O propósito deste trabalho é avaliar o impacto na adaptação do repositório DSPace para atuar como um cliente do serviço de recomendação.

Houve uma certa dificuldade em termos da própria inclusão de funcionalidades no DSpace. Mas após a identificação dos passos para esta customização, a inclusão das funções de registro de acesso e apresentação da recomendação foram bastante facilitadas. Neste sentido, considera-se que o uso de um Web service para acesso ao sistema de recomendação resultou efetivamente na redução da complexidade de sua integração com o repositório DSpace.

Em termos de testes, foram realizados medidas de tempo de resposta, para avaliar os atrasos de registro de acesso a recursos, e de apresentação da recomendação. No cenário de teste, tanto o lado cliente, como o servidor de recomendação estavam localizada na rede UFSC, sendo que o atraso de ida-e-volta de rede é inferior a 1ms.

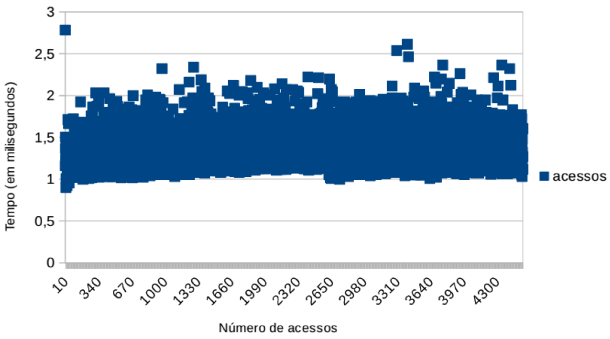

Figura 18. Tempo de registro de acesso ao recurso no sistema de recomendação.

Para o registro de acessos, foram simulados 4500 acessos a recursos. O atraso médio de registro destes acessos foi de 1,34s (com desvio padrão de 0,23) apresentados no gráfico da figura 18. Dado que o registro de acesso é assíncrono, este atraso não compromete a usabilidade do repositório. Mesmo assim, como este tempo é inferior a 2s, tempos limites de atrasos convencional da Web, conforme estudo feito por [Nah 2004], o resultado considera-se satisfatório. Conforme análise durante os testes, o componente que resultou em um maior uso de recursos de CPU e memória foi o próprio Tomcat. O Neo4J limitou-se a 1% de uso de CPU. Possíveis atualizações do Tomcat, e de configuração para melhor desempenho, poderão melhorar estes tempos, visto que a configuração utilizada foi a padrão.

Em termos de tempo de resposta da apresentação, foram simuladas 100 recomendações. Na recomendação por popularidade, o tempo médio de resposta foi de 0,64s (Desvio Padrão de 0,13) como apresentado no gráfico da figura 19. Já na recomendação por conteúdo, foi realizada inicialmente a simulação de 5 recursos por parte do usuário, e em seguida foram realizadas as medidas do tempo de resposta de 100 recomendações. Neste caso, o atraso médio foi de 0,63s (Desvio Padrão de 0,12) como visto no gráfico da figura 20.

Normalmente, devido à complexidade do processo da recomendação, o atraso da recomendação por conteúdo deveria ser maior que o tempo por popularidade. No caso dos experimentos, como o histórico de acessos do usuário foi pequeno, os tempos de repostas dos tipos de recomendação foram similares.

1.19. Conclusão

Até hoje não há solução disponível de sistema de recomendação para DSpace e sua implementação se torna importante visto que mostra-se ser uma grande aliado na localização rápida de conteúdos do interesse do usuário. Mas ao mesmo tempo, uma solução para estes sistemas não pode ser de difícil integração, haja visto que maior parte das organizações não dispõem de equipe especializada no DSpace.

Este trabalhou ocupou-se em estudar meios de uma implementação efetiva de um cliente Web service dentro da plataforma para repositórios digitais DSpace, além de apre-

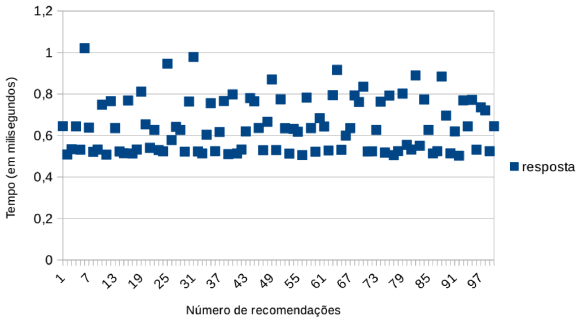

Figura 19. Tempo de resposta da apresentação baseado em recomendação por popularidade.

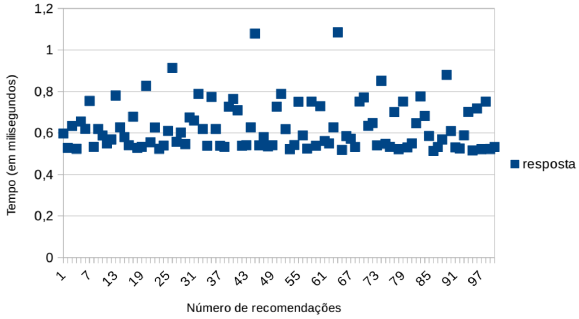

Figura 20. Tempo de resposta da apresentação baseado em recomendação baseada em conteúdo.

sentar conceitos e definições responsáveis pela fundamentação teórica do referente trabalho. O sistema de recomendação proposto foi implantando com sucesso na plataforma DSpace cumprindo seu objetivo de ser fracamente acoplável e adaptável ao domínio dos conteúdos de um RD, fato este que justifica pelo seu padrão Web service de projeto.

Apesar de relativamente simples, para se chegar ao estado final deste presente trabalho, foi realizado um grandioso esforço de reconhecimento do código-fonte e funcionamento do DSpace como um todo. Esforço que foi necessário devido a sua documentação escassa e não clara juntamente com uma comunidade de desenvolvedores ainda escondida no que se trata de de personalização e customização do código-fonte. O código-fonte é pouco documentado mostrando-se útil somente para quem fez parte de seu projeto de implementação. O fato do DSpace possuir muitos softwares de terceiros amarrados em si para seu funcionamento dificulta ainda mais para manter o sistema estável antes e após a alteração no código-fonte.

1.20. Trabalhos futuros

Para alguns dos trabalhos futuros, podemos citar:

- Teste do sistema de recomendação com itens e usuários reais;
- Otimização e correção de eventuais bugs do código-fonte criado;
- Implementação do sistema de recomendação no sistema em produção;
- Mudança da exibição da lista de recomendação criada para recomendações baseadas em conteúdo da página inicial para a página de recurso.

Referências

Anjorin, M., Dackiewicz, I., Fernández, A., Rensing, C., et al. (2012). A framework for cross-platform graph-based recommendations for tel. In *Proceedings of the 2nd workshop on recommender systems in technology enhanced learning*, pages 83–88.

Arellano, M. A. (2004). Preservação de documentos digitais. *Ci. Inf., Brasília*, 33(2):15–27.

Beham, G., Kump, B., Ley, T., and Lindstaedt, S. (2010). Recommending knowledgeable people in a work-integrated learning system. *Procedia Computer Science*, 1(2):2783–2792.

Casagrande, M. F. R. (2014). Técnica de recomendação para repositórios digitais baseada em metadados e agrupamento de usuários.

Castagné, M. (2013). Institutional repository software comparison: Dspace, eprints, digital commons, islandora and hydra.

DCMI, D. C. M. I. et al. (2012). Dublin core metadata element set, version 1.1.

de Oliveira, R. R. and de Carvalho, C. L. (2011). Bibliotecas digitais e o repositório fedora.

DSpace (2016a). Dspace - dspace is a turnkey institutional repository application.

DSpace, D. T. (2016b). Dspace 5.x documentation.

Duraspace, C. Xmlui configuration and customization.

Elliott, D., Rutherford, J., and Erickson, J. (2008). A recommender system for the dspace open repository platform. Technical report, Tech. Rep., HP Laboratories, Bristol.

Group, W. O. W. (2012). Owl 2 web ontology language document overview (second edition).

Heery, R. and Anderson, S. (2005). Digital repositories review.

IBICT, I. B. d. I. e. C. e. T. (2016). Sobre repositórios digitais.

Lemos, F. D., Carmo, R. A., Viana, W., and Andrade, R. (2012). Improving photo recommendation with context awareness. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, pages 321–330. ACM.

Nah, F. F.-H. (2004). A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163.

Phillips, S., Green, C., Leggett, J., Maslov, A., Mikeal, A., and Surratt, B. (2005). Manakin.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2011). *Introduction to recommender systems handbook*. Springer.

Salles, A. and Willrich, R. (2015). Recommending web service based on ontologies for digital repositories. In *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*, pages 65–72. ACM.

Shintaku, M. and Meirelles, R. F. (2010). Manual do dspace: administração de repositórios.

Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., Tansley, R., and Walker, J. H. (2003). Dspace: An open source dynamic digital repository.

Tansley, R., Bass, M., Stuve, D., Branschofsky, M., Chudnov, D., McClellan, G., and Smith, M. (2003). The dspace institutional digital repository system: current functionality. In *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, pages 87–97. IEEE Computer Society.

Wolf, A. S., Monteiro, A. P. L., and Valmorbida, W. (2013). Biblioteca digital da univates utilizando o software dspace. *Destaques Acadêmicos*, 1(4).